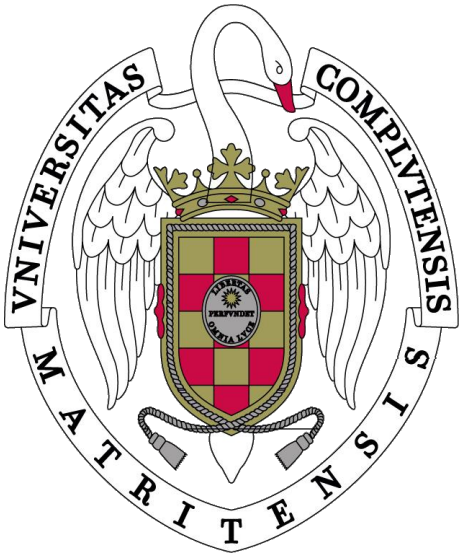


SISTEMAS INFORMÁTICOS CURSO 2006-2007

ORFEO

Ignacio Ferrando García
Luis Molero Blázquez
Carlos Rioboo Crespo

Dirigido por el profesor Jaime Sánchez Hernández

Facultad de Informática Universidad Complutense de Madrid

Resumen

El proyecto ha consistido en el desarrollo de una aplicación que ayuda a realizar tareas musicales de composición en un entorno gráfico multipista, automatizando ciertos procedimientos y facilitando otros, haciendo posible ver y escuchar el resultado de ideas musicales de una manera rápida, trabajando con una notación textual.

La aplicación ofrece procedimientos de creación y modificación de fragmentos musicales que han sido ampliamente usados en la composición desde el Barroco: retrogradaciones, transposiciones, inversiones, añadir intervalos, disminuciones, aumentaciones, cambios de dirección, modulaciones, arpegios, cambios rítmicos, marchas tonales y cromáticas... Además ofrece la posibilidad de generar progresiones armónicas, ritmos, percusiones y acordes.

Abstract

The project has consisted of the development of a tool of aid to the composition, in a multitrack graphic user interface, making several procedures automatic and facilitating others. The application makes possible to listen and to view the result of musical ideas, easy and quicly, working with textual notation.

Orfeo offers procedures to create and modify musical fragments, that have been widely used since the Baroque: retrogradation, transposition, change of direction, inversion, broken chords, interval addition, diminution, augmentation, modulations, rithm changes, tonal and chromatic marchs... Furthermore Orfeo cans generate armomic progressions, rithms, percussions and chords.

Lista de palabras para búsqueda bibliográfica:

música, composición, orfeo, abc, midi, pianola

ÍNDICE

1.INTRODUCCIÓN

1.1.¿Qué es Orfeo?

Orfeo es una aplicación que ayuda a realizar tareas musicales de composición, automatizando ciertos procedimientos y facilitando otros, haciendo posible ver y escuchar el resultado de ideas musicales de una manera rápida, trabajando con una notación basada en un lenguaje de notación musical textual llamado ABC.

Orfeo ofrece procedimientos de creación y modificación de fragmentos musicales que han sido ampliamente usados en la composición desde el Barroco.

Exige un proceso de familiarización con el lenguaje ABC y la sintaxis específica de Orfeo para sacar verdadero provecho de la aplicación. Así mismo el usuario debe tener conocimientos de música para poder manejar con soltura las todas las posibilidades que ofrece Orfeo.

Posee 18 pistas. A cada una de estas pistas se le puede asociar un instrumento midi (a elegir entre 128) y un volumen de reproducción (en el rango pp-ff). Las pistas 17 y 18 están destinadas especialmente a la percusión. Se reproducen a través del canal 10 midi en el que cada nota midi tiene asociada un sonido de percusión. En estas dos pistas no se asocia el instrumento sino un kit de percusión.

Trabajamos siempre sobre una única pista pero hay acciones que realizan cambios en todas las pistas para mantener coherencia armónica, de compás y tempo. Estas acciones son: cambio de compás, cambio de escala, cambio de tonalidad y cambio de tempo.

Cada pista tiene un campo Out que cuando está seleccionado hace que el contenido de la pista se vuelque a la salida del archivo midi si se ha elegido la opción reproducir o al archivo ps si se ha elegido ver partitura. Sólo las pistas marcadas se exportan cuando realizamos la acción de exportar.

El uso de la interfaz principal y las pistas se explica en el capítulo 3.2.

La pianola muestra en formato rodillo de piano el contenido de la pista seleccionada. El uso de la pianola se explica en el capítulo 3.2.9.

El nombre es un homenaje a la Ópera “Orfeo” de Monteverdi, considerada la primera ópera y de cuyo estreno se cumple en 2007 el cuarto centenario.

Actualmente no tenemos conocimiento de ninguna aplicación que realice todas las tareas de Orfeo y que pueda ser configurable y ampliable de forma tan fácil usando notación textual.

1.2.¿Qué es ABC?

ABC es una notación musical en modo texto que permite representar prácticamente la totalidad de la notación musical occidental. Es una notación con varias versiones en las que si bien hay un alto porcentaje de coincidencia no hay un único estándar. Existen varias aplicaciones que permiten convertir un documento en ABC a otros formatos que nos permitan ver la representación en partitura y escucharlo en formato Midi. El hecho de que existan distintas versiones de ABC hace que las aplicaciones que traducen ABC a PostScript o Midi ofrezcan resultados distintos dependiendo de la versión ABC utilizada.

El lenguaje que utiliza Orfeo para editar fragmentos musicales es un subconjunto de ABC. De esta forma podemos visualizar y escuchar la música creada con Orfeo, usando las aplicaciones que traducen ABC a PostScript y Midi.

1.3.¿Qué es MIDI?

MIDI es el acrónimo de *Musical Instrument Digital Interface* (Interfaz Digital de Instrumentos Musicales). Se trata de un protocolo industrial estándar que permite a las computadoras, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos comunicarse y compartir información para la generación de sonidos.

En el capítulo 9.1 se puede consultar el Mapa de instrumentos General Mid.i
En la capítulo 9.2 se puede consultar el Mapa General Midi de instrumentos de percusión.
En la capítulo 9.3 se pueden consultar los Kits de Percusión General Standard.
En la capítulo 9.4 se puede consultar la relación de notas MIDI y frecuencias.

Esta información es de utilidad para los usuarios de la aplicación debido a que Orfeo hace uso de estos recursos Midi.

1.4.¿Qué es PostScript?

PostScript es un Lenguaje de Descripción de Página , utillizado en muchas impresoras y como formato de transporte de archivos gráficos en talleres de impresión profesional.

Ghostscript es una implementación abierta de un intérprete compatible con PostScript. Orfeo usa Ghostscript para mostrar en partitura la música generada.

1.4.Capítulos de la Memoria

El capítulo 2 está dedicado a explicar los lenguajes que emplea Orfeo para generar y editar fragmentos musicales. En el capítulo 3 se explica cómo ejecutar y usar la aplicación. El capítulo 4 está dedicado a aspectos de implementación. Las herramientas utilizadas para crear y usar Orfeo se pueden consultar en el capítulo 5. En el capítulo 6 aparecen las principales dificultadas halladas en la realización del proyecto. En el capítulo 7 se comentan los objetivos alcanzados y posibles ampliaciones de la aplicación. La organización interna de los miembros del grupo para la elaboración del proyecto se encuentra en el capítulo 8. El capítulo 9 es un apéndice con información sobre midi. Finalmente, el capítulo 10 alberga la bibliografía consultada para la elaboración del proyecto.

2.Los lenguajes de Orfeo

En este capítulo se exponen los diferentes lenguajes que emplea la aplicación para editar y crear música.

2.1.Los lenguajes que maneja Orfeo

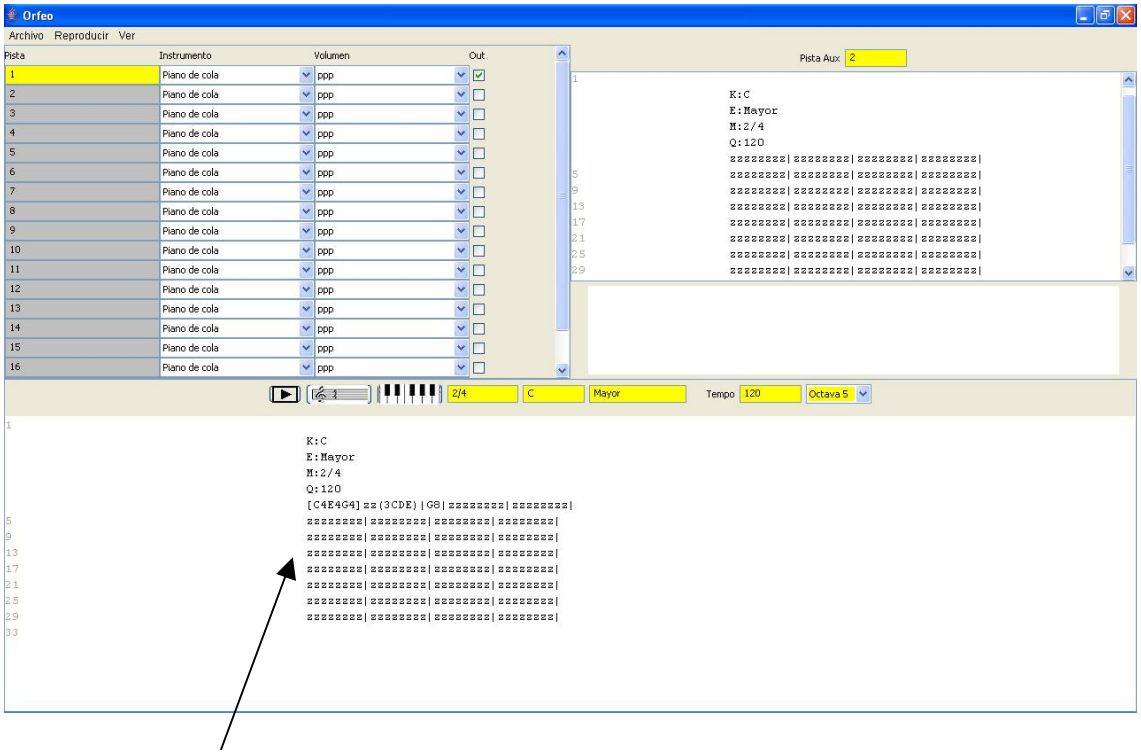
Orfeo maneja cinco lenguajes:

- Lenguaje de edición, basado en ABC.
- Lenguaje de las Progresiones Armónicas, para escribir progresiones en formato texto que Orfeo pueda convertir en música.
- Lenguaje de los Ritmos, que permite escribir ritmos en formato texto que Orfeo puede convertir en música.
- Lenguaje de las Percusiones , para escribir ritmos con percusiones asociadas en formato texto que Orfeo pueda convertir en música.
- Lenguaje de los Acordes, que permite escribir acordes en formato texto que Orfeo puede convertir en música.

2.2.Lenguaje de Edición

Este lenguaje es el que el usuario debe manejar para poder crear, editar y aplicar transformaciones musicales empleando el campo de edición.

Está basado en un subconjunto de ABC al que se le han añadido dos particularidades:
-Instrucción para indicar la escala con la que se trabaja.
-Modificación de la notación para dosillos y tresillos para facilitar el trabajo de los traductores.

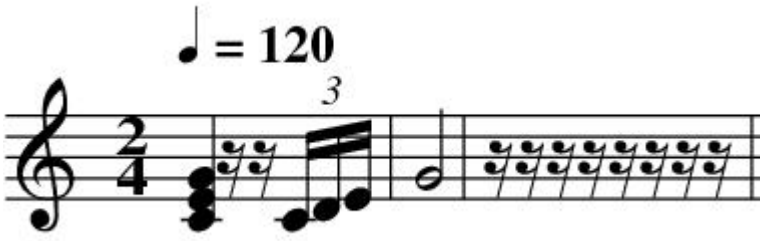


Panel de Edición

La información que aparece el panel de edición está escrita con el lenguaje de Orfeo para edición:

K:C
E:Mayor
M:2/4
Q:120
[C4E4G4]zz(3CDE)|G8|zzzzzzzz|

En formato de partitura se obtiene:



Puesto que se hace uso del protocolo midi se usa un sistema enarmónico.

En un sistema enarmónico dos notas se llaman enarmónicas cuando, a pesar de tener distinto nombre, tienen igual sonido (frecuencia sonora).

Por ejemplo: las notas *sol* sostenido y *la* bemol son enarmónicas. Si utilizáramos otros sistemas las notas sol sostenido y la bemol tendrían distintos sonidos.

Por lo tanto Orfeo mostrará sol sostenido o la bemol dependiendo de la tonalidad en la que estemos trabajando. Lo mismo ocurrirá para el resto de notas enarmónicas. Esta consideración permite ahorrar opciones en el menú de tonalidades, disponiendo de las siguientes posibilidades:

Tonalidad={ C, C#, D, Eb, E, F, F#, G, Ab, A, Bb, B}

La tonalidad se indica con la instrucción K:Tonalidad

Determina la tónica. El modo se determina eligiendo la escala:

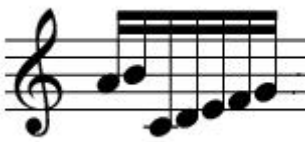
Escala={Mayor, MixtaI, MixtaII, MenorNatural, Armónica, Melódica}
La escala se indica con la instrucción E:Escala

Las notas se representan con notación anglosajona:

Notación Anglosajona	A	B	C	D	E	F	G
Notación Española	La	Si	Do	Re	Mi	Fa	Sol

Ejemplo:

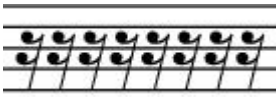
ABCDEFG



Los silencios se representan con z. Si se dejan compases incompletos la aplicación los completa con los silencios que hagan falta.

Ejemplo:

z z z z z z z z



La extensión corresponde a los 128 notas midi.

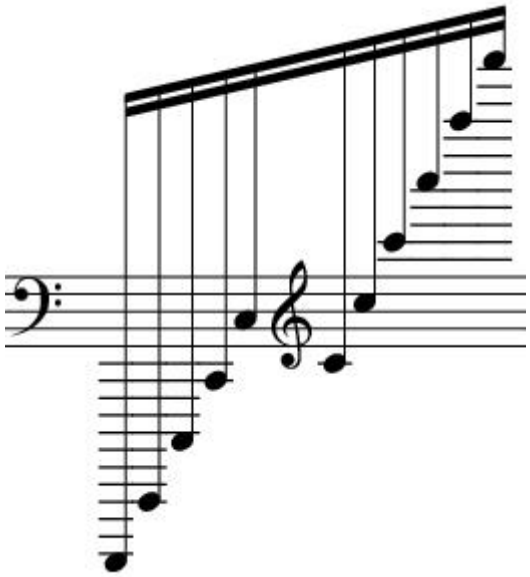
En modo texto la extensión es C,,,,, - g^''''.

El Do central del piano (261.626 Hz) se representa con C y corresponde a la nota midi número 60 (en midi C5) . Cada octava inferior se indica con ",". Así el Do3 se escribe como C,,.

El Do6 se escribe como c. Cada octava adicional se indica con " ' ". Así el DO8 se escribe como c''.

Sonido	Notación
C0	C,,,,,
C1	C,,,,
C2	C,,,
C3	C,,
C4	C,
C5	C
C6	C
C7	c'
C8	c''
C9	c'''
C10	c''''

C,,,,,C,,,,C,,,C,,C,Ccc'c''c'''c''''



Para indicar alteraciones debemos usar caracteres distintos a los usados en la notación musical tradicional:

Sostenido	^
Bemol	_
Becuadro	=

Las alteraciones tienen validez en todo el compás.

Ejemplo:

F^F_A=F

Al usar un sistema enarmónico y para facilitar la escritura, ORFEO no permite el doble sostenido ni el doble bemol.

Los compases disponibles y sus duraciones en semicorcheas son:

Compás	Duración
2/4	8
¾	12
4/4	16
3/8	6
6/8	12
9/8	18
12/8	24

La instrucción utilizada para indicar el compás es: M:TipoCompás

Ejemplo:
M:2/4
La instrucción utilizada para indicar el tempo es: Q:NNM, siendo NNM el número de negras por minuto.

Ejemplo:

Q:120

Indica un tempo a 120 negras por minuto.

Duraciones:

La medida de referencia es la semicorchea.

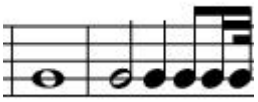
Las figuras permitidas y sus valores en semicorcheas son:

Figura	Valor
Redonda	16
Blanca	8
Negra	4
Corchea	2
Semicorchea	1
Fusa	0.5

Para simplificar, cuando se quiere asociar a una nota una duración de valor 1, se omite la duración y cuando se quiere escribir una fusa se añade el carácter '/' detrás del nombre de la nota. Para las corcheas se añade un 2, negras un 4, blancas un 8 y redondas 16. Se puede escribir cualquier valor entero entre 1 y 16.

Ejemplo:

```
|G16|G8G4G2GG/
```



Para obtener valores no enteros se debe usar la ligadura de ABC '-' seguido de la nota a ligar y el carácter'/'.

Ejemplo:

```
G-G/
```

Es posible escribir dosillos y tresillos de la siguiente forma:

Dosillos→ (2 NotasDosillo)

Tresillos→ (3 NotasTresillo)

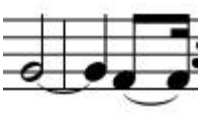
Ejemplo:

```
(3G2A2B2)
```

Para ligar notas o acordes se usa el carácter '-':

Ejemplo:

```
G8-|G4F2-F
```



Para escribir un acorde las notas que formen el acorde deben escribirse con sus respectivas duraciones entre los caracteres '[' ']'

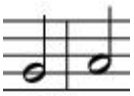
Ejemplo:

[C4E4G4]

Las divisiones de compás se indican con el carácter '|'

Ejemplo:

C8|E8|



Los cambios de Tonalidad, Escala, Compás y Tempo deben ir cada uno en una línea distinta y para evitar errores deben realizarse a través de las opciones que aparecen en el menú.

El lenguaje no permite:

-Escribir distintas voces al modo ABC en una misma pista, pero si permite escribir acordes.

-Instrucciones ABC de carácter dinámico como (pp,mf...) o expresivo como el stacatto, los acentos o las ligaduras expresivas.

-Notas de adorno.

-Grupos de notas distintos a los dosillos y tresillos.

2.3.Lenguaje de las Progresiones Armónicas.

Este lenguaje permite escribir progresiones armónicas formadas por acordes con sus respectivas duraciones.

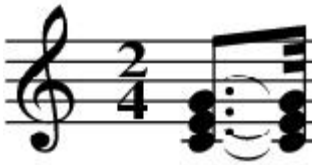
Cada elemento de la progresión debe ser de la forma:

Grado = I, II, III, IV, V, VI, VII.
Indica el grado dentro de la tonalidad de referencia.

Duración: Fusa, notada por / (equivale a media unidad de duración).
Resto de valores, un entero que indica el número de semicorcheas que dura el acorde. Si se desea un valor no entero (por ejemplo 3 corcheas y media) añadimos al entero el carácter L.

Ejemplo:
En un compás de 2/4 tomando como referencia la tonalidad de Do se obtendría:

Lenguaje de progresiones: I3L
Lenguaje de edición: [C3E3G3]-[C/E/G/]



Los elementos se separan con el carácter '-'

Ejemplo:
En un compás de 2/4 tomando como referencia la tonalidad de Do se obtendría:

I4-IV2-V2-I8

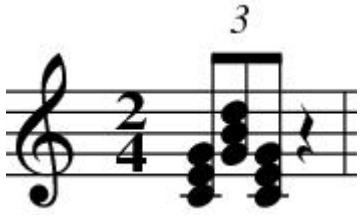
[C4E4G4][F2A2c2][G2B2d2]|[C8E8G8]|

Para incluir tresillos y dosillos escribimos los elementos que los forman entre paréntesis, con un 2 tras el paréntesis abierto si es un dosillo y un 3 si es un tresillo.

Ejemplo:
En un compás de 2/4 tomando como referencia la tonalidad de Do se obtendría:

(3I2-V2-I2)

(3[C2E2G2][G2B2d2][C2E2G2])



Dependiendo del compás puede no tener sentido aplicar ciertos patrones rítmicos con dosillos y tresillos.

Los lenguajes de las progresiones armónicas, ritmos, percusiones y acordes permiten la inclusión de comentarios para poder dar un nombre o informar sobre el compás para el que es más conveniente. Los comentarios se incluyen añadiendo ':' seguido del texto que deseemos siempre y cuando no exceda una línea.

Ejemplo:

I4-IV2-V2-I8: Progresión Típica 2/4

2.4.Lenguaje de los ritmos

Permite escribir estructuras rítmicas genéricas, que pueden utilizarse en cualquier compás.

Los elementos rítmicos deben estar formados por una N seguida de la duración (indicada de la misma manera que con las progresiones armónicas) o por una Z seguida de la duración si se quiere expresar un silencio.
Los elementos formados con N generan notas que son la tónica de la tonalidad de referencia.

Los elementos rítmicos deber ir separados por '-'

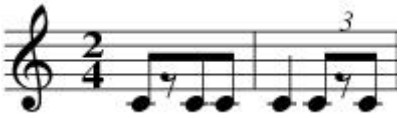
Para incluir tresillos y dosillos se emplea el mismo procedimiento que el usado para las progresiones armónicas.

Ejemplo:

N2-Z2-N2-N2-N4-(3N2-Z2-N2)

En un compás de 2/4 tomando como referencia la tonalidad de Do se obtendría:

C2z2C2C2|C4(3C2z2C2)



2.5.Lenguaje de las percusiones

La notación es similar a la de los ritmos pero en este caso en vez de utilizar la letra N se escribe el número midi asociado al sonido de percusión que queremos que suene y lo separamos de la duración con el carácter ’,’.

Los sonidos asociados a cada número se pueden consultar en el capítulo 9.3.

Ejemplo:

38,2-38,2-Z2-38,2-49,4

El 38 tiene asociado el sonido de Acoustic snare.
El 49 tiene asociado el sonido de Crash Cymbal 1.

En un compás de 3/4 se obtendría:

D,,2 D,,2z2 D,,2^C,4

2.6.Lenguaje de los acordes

Permite escribir cualquier acorde de una manera rápida e intuitiva.

Se escribe el grado de la escala a partir del cual se quiere formar el acorde, seguido de las distancias en semitonos que hay que sumar al grado para obtener las notas del acorde.

Grado(D0,D1,..,Dn)

Grado = I, II, III, IV, V, VI, VII

(D0,D1,..,Dn) donde cada Di es un entero que indica cuánto hay que sumar a la fundamental (nota midi asociada al Grado en la tonalidad de referencia, teniendo en cuenta la octava actual) para ir obteniendo el acorde. El acorde tendrá n notas.

Ejemplo:

I(0,4,7), sería el acorde fundamental de Do si está trabajando en la tonalidad de Do. Con la octava 5 seleccionada la nota asociada al grado I en la tonalidad de Do es la nota midi 60. Para obtener el acorde sumamos 0,4 y 7, obteniendo un acorde formado por las notas 60,64 y 67.

[CEG]

3.GUÍA DE USO

En este capítulo se explica cómo usar la aplicación con todas las opciones disponibles.

3.1.Ejecutar Orfeo

Se debe tener instalado Ghostscript, GSview ,Timidity y la versión 1.5 del JDK de java. Estos componentes se proporcionan con orfeo pero también pueden encontrarse en los en los sitios indicados en la bibliografía en el capítulo 10.

Para ejecutar la aplicación simplemente basta con hacer doble clic en el archivo Orfeo.jar.

En el directorio raíz de Orfeo encontramos los archivos:

Orfeo.jar, tuprolog.jar

Y las carpetas:

- **acordes:** contiene el archivo acordes.txt con los acordes que Orfeo carga por defecto.

- **abcm2ps:** contiene la aplicación abcm2ps.exe que permite traducir la música que genera Orfeo a un formato de partitura en formato postscript(.ps). En esta carpeta también encontramos la aplicación Gsview que permite visualizar los archivos .ps .

- **abc2midi:** contiene la aplicación abc2midi.exe que permite traducir la música que genera Orfeo a formato midi.

- **icons:** contiene los archivos .jpg que utiliza la aplicación como iconos en algunos botones.

- **opus:** contiene composiciones de prueba y es el directorio por defecto para abrir y guardar archivos.

- **percusiones:** contiene el archivo percusiones.txt con las percusiones que Orfeo carga por defecto.

- **prolog:** contiene el archivo orfeoProlog.pl que contiene el código prolog que utiliza la aplicación.

- **progresiones:** contiene el archivo progresiones.txt con las progresiones que Orfeo carga por defecto.

- **ritmos:** contiene el archivo ritmos.txt con los ritmos que Orfeo carga por defecto.

- **temp:** carpeta destinada a almacenar los archivos temporales que se crean al utilizar abcm2ps y abc2midi

- **timidity:** contiene la aplicación Timidity que permite escuchar la música generada por Orfeo.

*La versión actual es ejecutable únicamente en sistemas Windows, debido a que en este momento sólo se han incluido las aplicaciones auxiliares en su versión para Windows.

* Para poder ejecutar Orfeo existen las siguientes alternativas:

1)Ir al directorio donde esté instalada la máquina virtual de java de la versión 1.5 del JDKy en línea de comandos teclear:

C:\Java\JDK1.5\bin> java –jar C:\ORFEO\Orfeo.jar

Suponiendo que el directorio donde se encuentra la aplicación java.exe sea C:\Java\JDK1.5\bin y que el directorio donde se encuentra Orfeo sea C:\ORFEO.

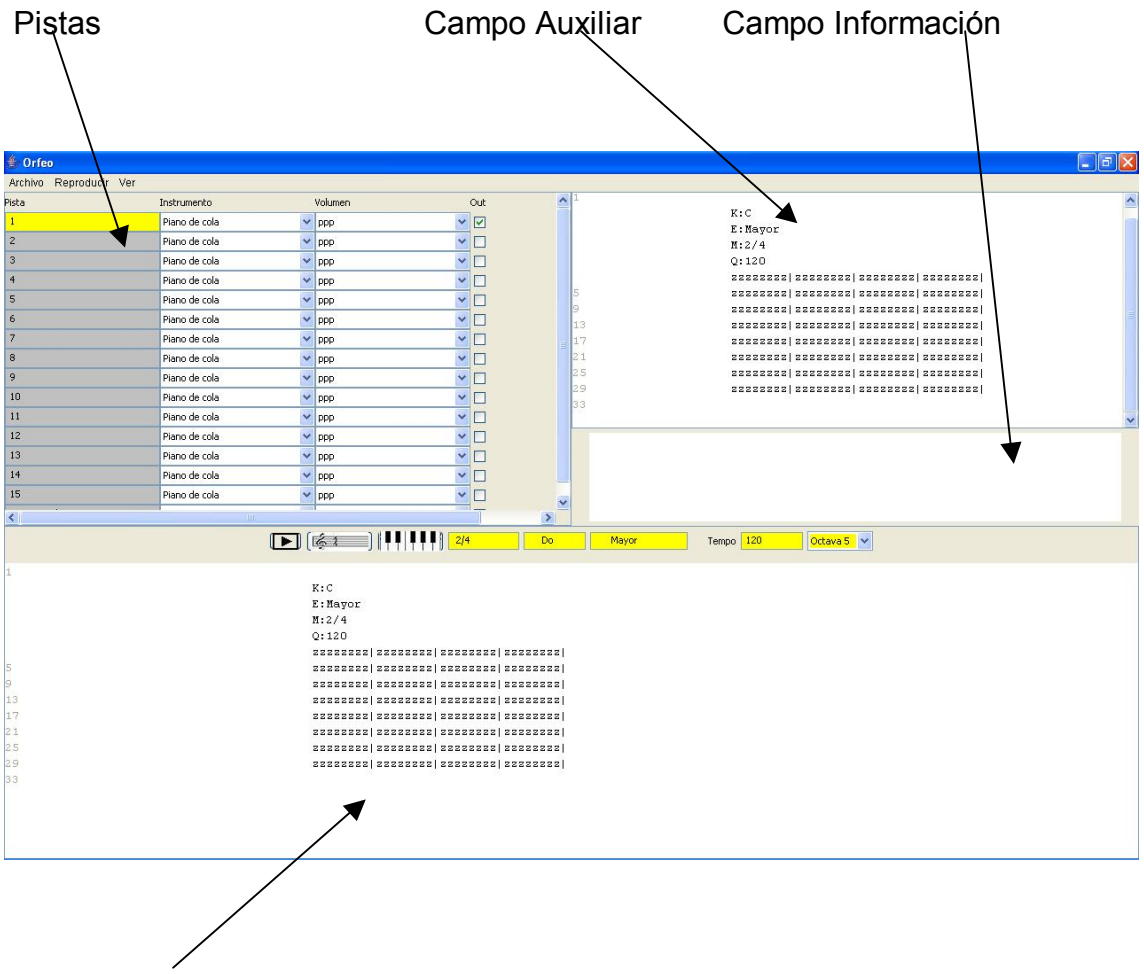
2)Ir al directorio donde esté la carpeta ORFEO que contiene todo lo necesario para ejecutar la aplicación y en línea de comandos teclear:

C:\ORFEO>"C:\Java\JDK1.5\bin\java" –jar Orfeo.jar

Suponiendo que el directorio donde se encuentra la aplicación java.exe de la versión JDK1.5 sea C:\Java\JDK\1.5\bin y que el directorio donde se encuentra Orfeo sea C:\ORFEO.

3.2.Usando Orfeo

Al ejecutar la aplicación aparece la siguiente ventana que corresponde a la interfaz principal.



Panel de Edición

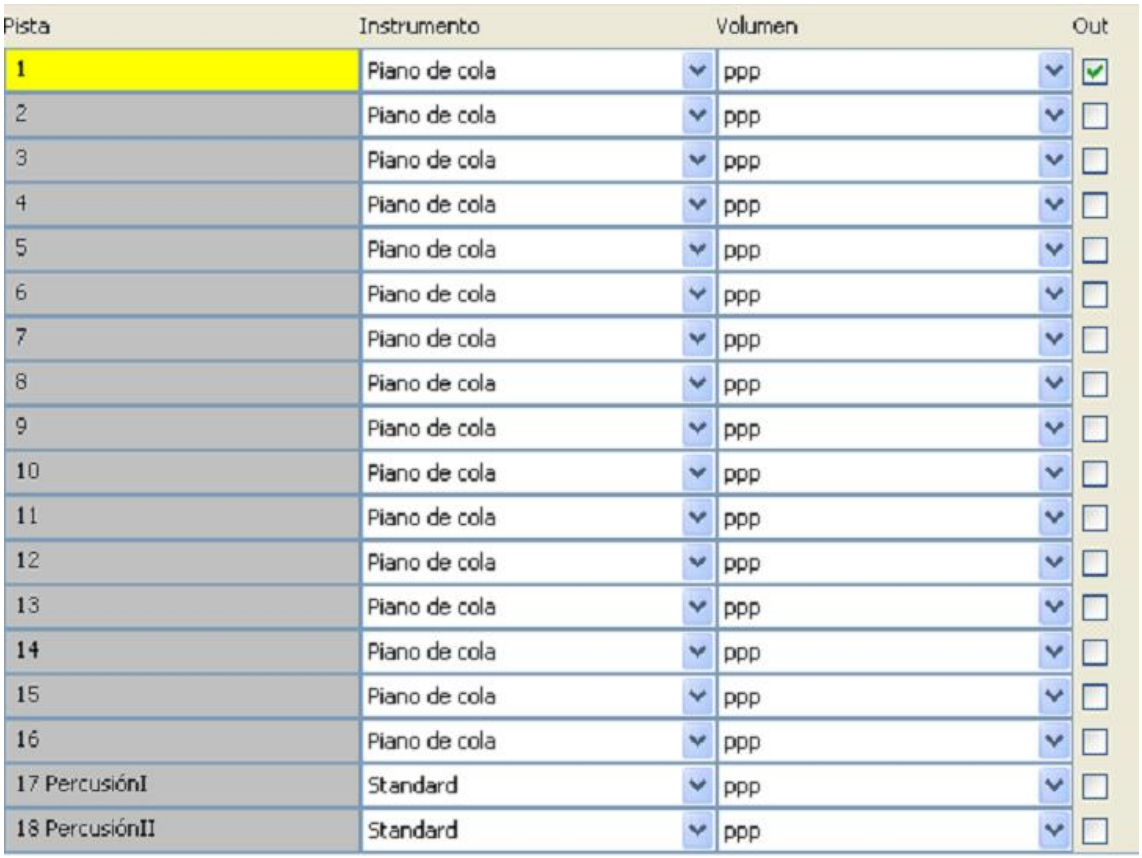
Se distinguen tres áreas con funciones diferentes:

-Panel de Pistas: donde se encuentran las 18 pistas disponibles.

-Panel de edición: donde se crea y modifica la música.

-Panel auxiliar: para tareas auxiliares e informativas.

3.2.1.Panel de Pistas



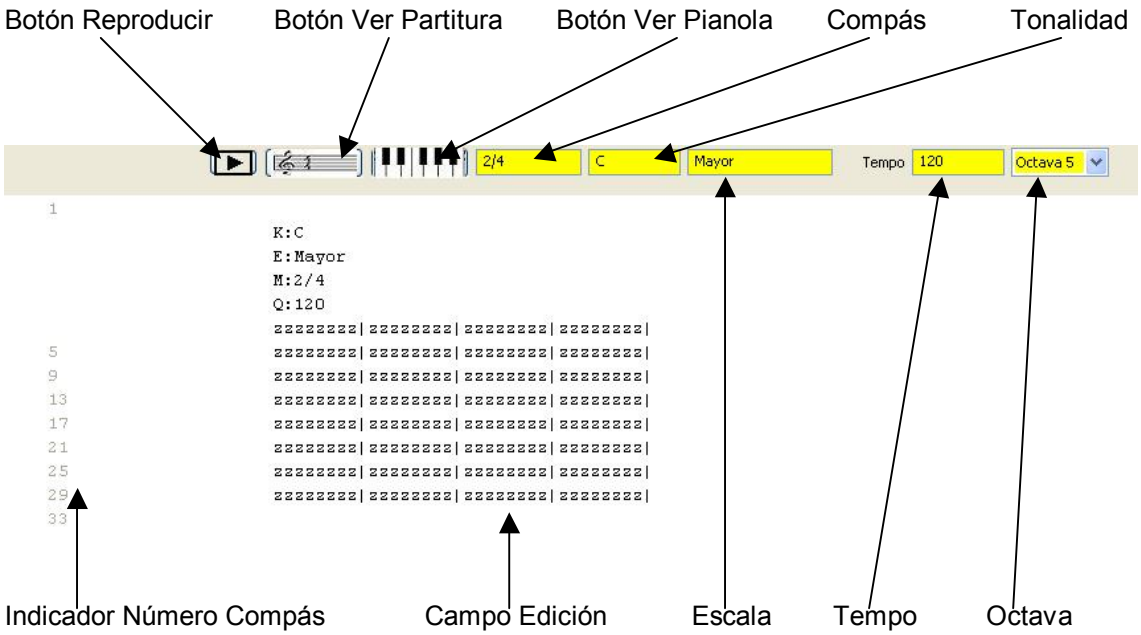
Pista: La pista seleccionada haciendo clic se reconoce por cambiar su color de gris a amarillo. El contenido de la pista seleccionada se muestra en el Campo Edición.

Instrumento: asocia un instrumento a la pista. Si es la pista 17 o 18 se asocia un ambiente de percusión.

Volumen: asocia el volumen de reproducción a la pista.

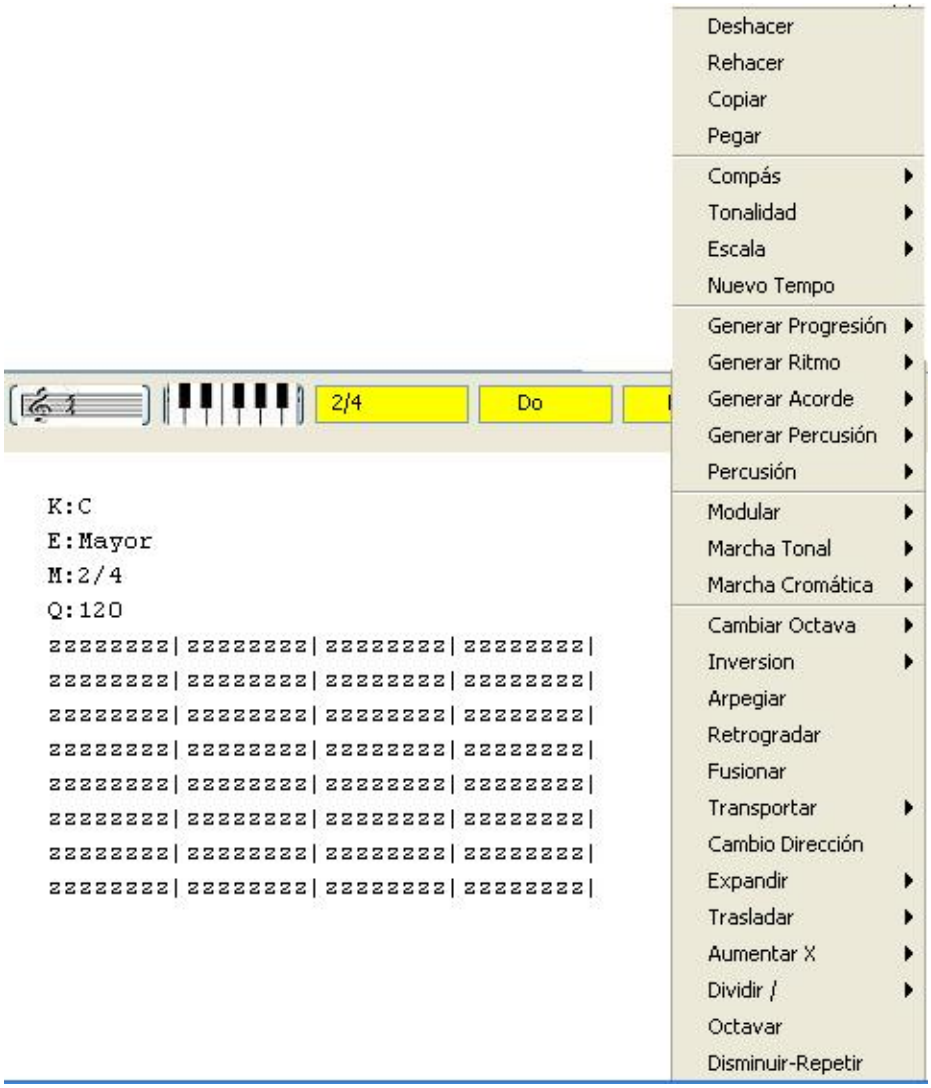
Out: sólo las pistas con esta casilla marcada son las que se reproduce, visualizan y exportan.

3.2.2.Panel de Edición



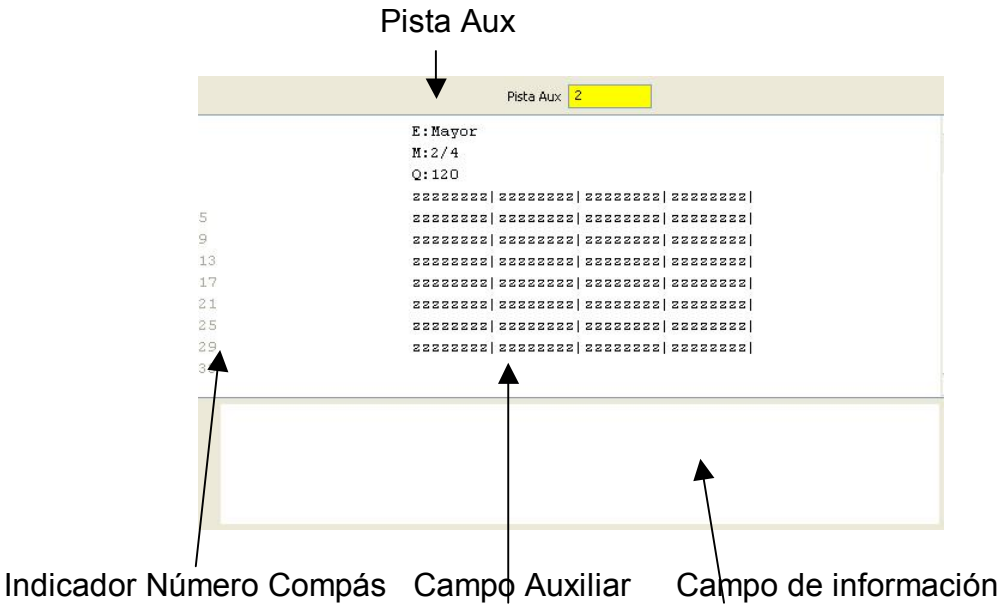
- **Indicador Número Compás:** sirve para llevar control de los compases, facilitando su localización.
- **Botón Reproducir:** reproduce las pistas seleccionadas (campo Out seleccionado) .
- **Botón Ver Partitura:** muestra en partitura las pista seleccionadas.
- **Botón Ver Pianola:** muestra la pianola. Si la pista contiene dosillos o tresillos no se puede mostrar.
- **Compás:** indica el compás.
- **Tonalidad:** indica la tonalidad.
- **Escala:** indica la escala.
- **Tempo:** indica el tempo.
- **Octava:** indica la octava que se utiliza para las opciones de Generar Progresión, Generar Ritmo y Generar Acorde.

Al hacer clic en el campo de edición se detecta el compás, la tonalidad, la escala y el tempo del compás en el que se sitúa el cursor y se muestran en los correspondientes campos .



Menú desplegable con las opciones de edición y aplicación de procedimientos para la creación y modificación de fragmentos musicales. El menú se despliega con al hacer clic en el botón derecho del ratón, en el campo de edición.

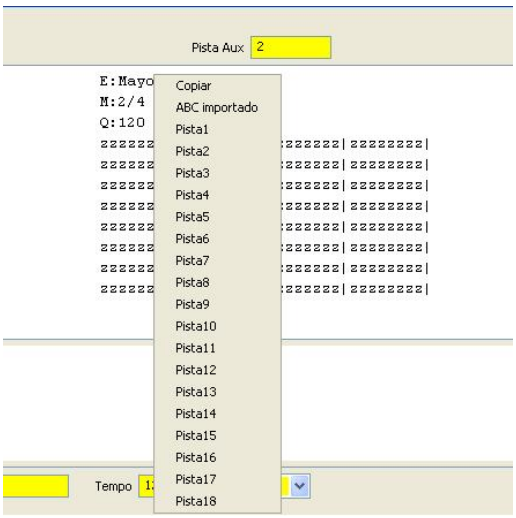
3.2.3.Panel Auxiliar



- **Campo Auxiliar:** muestra el contenido de una pista de un archivo ABC importado.

- **Campo de Información:** Cuadro de información: informa de los fallos en la sintaxis cuando se intenta aplicar un cambio de compás, tonalidad, escala, tempo, se realiza un procedimiento compositivo o se intenta mostrar la partitura o la pianola y reproducir la obra.

- **Pista Aux:** muestra el número de pista que se está mostrando en el campo auxiliar si se ha seleccionado una pista o un archivo ABC si se ha seleccionado la opción ABC importado.



Menú desplegable que aparece al hacer click con el botón derecho en el campo auxiliar.

3.2.4.Opciones de los menús Archivo, Reproducir y Ver

Menú Archivo:

- Nuevo: reinicia el contenido de las pistas.
- Abrir: abre un archivo en formato Orfeo con extensión .orf
- Guardar como: guarda el trabajo como archivo Orfeo con extensión .orf
- Guardar: guarda el trabajo como archivo Orfeo con extensión .orf
- Cargar Progresiones: carga desde un archivo progresiones, que se añaden al menú Generar Progresión. El fichero debe tener una progresión por línea. Cada progresión debe estar formada por al menos un acorde.
- Cargar Ritmo: carga desde un archivo de ritmos, que se añaden al menú Generar Ritmo. El fichero debe tener un patrón rítmico por línea.
- Cargar Percusión: carga desde un archivo de percusiones, que se añaden al menú Generar Percusión. El fichero debe tener un patrón de percusión por línea.
- Cargar Acordes: carga desde un archivo ritmos, que se añaden al menú Generar Acorde. El fichero debe tener un acorde por línea.
- Importar ABC: importa un archivo en formato ABC para poder reutilizar en la aplicación.
- Exportar ABC: exporta las pistas seleccionadas en formato ABC
- Exportar Midi: exporta las pistas seleccionadas en formato MIDI
- Exportar PS: exporta las pistas seleccionadas en formato PS

Menú Reproducir:

- Play: reproduce las pistas seleccionadas, lanzando la ejecución de Timidity.

Menú Ver:

- Partitura: muestra en partitura las pistas seleccionadas.
- Pianola: muestra la pianola, si la pista no contiene dosillos ni tresillos.

3.2.5.Opciones de Edición

- **Deshacer:** deshace la última acción realizada (se considean acciones todas las opciones del menú desplegable del campo de edición). Cuando seleccionamos una pista el contenido de la pista en la que se estaba trabajando se guarda y pueden deshacerse los cambios mediante la opción Deshacer.

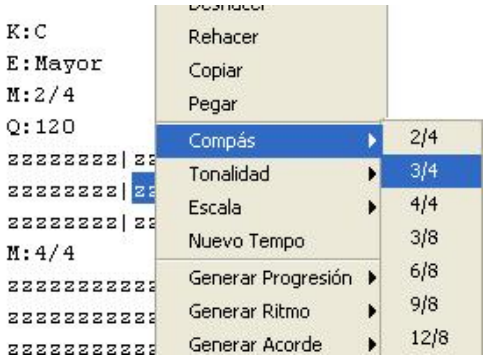
- **Rehacer:** rehace la última acción deshecha.

- **Copiar:** copia el fragmento seleccionado. También puede emplearse la combinación ctrl+c.

- **Pegar:** pega el fragmento seleccionado. También puede emplearse la combinación ctrl+v.

3.2.6.Opciones de Compás, Tonalidad, Escala y Tempo

- **Compás:** cambia el compás en todas las pistas desde el compás donde se haya hecho clic con el ratón, hasta donde haya otro cambio de compás o hasta el final si no hay ningún cambio de compás en el resto de las pista. Elimina el contenido si lo hubiera de los compases a los que afecta el cambio de compás.



Resultado

```
K:C
E:Mayor
M:2/4
Q:120
zzzzzzzz|zzzzzzzz|zzzzzzzz|zzzzzzzz|
zzzzzzzz|
M:3/4
zzzzzzzzzzzz|zzzzzzzzzzzz|zzzzzzzzzzzz|
zzzzzzzzzzzz|zzzzzzzzzzzz|
M:4/4
zzzzzzzzzzzzzzzz|zzzzzzzzzzzzzzzz|
zzzzzzzzzzzzzzzz|zzzzzzzzzzzzzzzz|zzzzzzzzzzzzzzzz|
```

- **Tonalidad:** cambia la tonalidad en todas las pistas desde el compás donde se haya hecho clic con el ratón, hasta donde haya otra tonalidad o hasta el final si no hay ningún cambio de tonalidad en el resto de las pista.

Resultado

```
K:C
E:Mayor
M:2/4
Q:120
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
K:G
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
K:D
[D8F8A8]|[A8c8e8]|[A8c8e8]|[D8F8A8]|
```

- **Escala:** cambia el compás en todas las pistas desde el compás donde se haya hecho clic con el ratón, hasta donde haya otra escala o hasta el final si no hay ningún cambio de escala en el resto de las pista.

Resultado

```
[D8F8A8]|A2c2e2A2|[A8c8e8]|D2F2A2D2|
E:Armónica
zzzzzzzz|zzzzzzzz|zzzzzzzz|zzzzzzzz|
```

- **Nuevo Tempo:** introduce el nuevo tempo indicado en el campo Tempo, desde el compás donde se haya hecho clic, hasta donde haya otro cambio de tempo o hasta el final de la obra si no hay cambio de tempo en la pista. La figura de referencia es la negra. El campo Tempo indica el número de negras por minuto.

Resultado

```
Q:120
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
Q:200
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
[D8F8A8]|A2c2e2A2|[A8c8e8]|D2F2A2D2|
```

3.2.7.Opciones de generación

- **Generar Progresión:** genera una progresión armónica teniendo en cuenta la tonalidad y la escala. Los cinco primeras opciones del menú Generar Progresión tienen duraciones en función de compases o medios compases. Tienen una duración total de cuatro compases. Se generan tantos compases como compases se hayan seleccionado con el ratón. Si no se selecciona nada se generan cuatro compases.

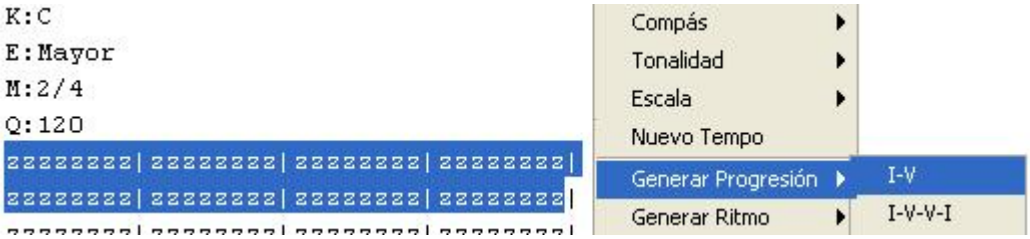
I-V: genera dos compases con la tónica y dos con la dominante.

I-V-V-I: genera un compás con la tónica, dos con la dominante y uno con la tónica.

I-V-I-V-I-V: genera medio compás con la tónica, medio compás con la dominante, uno con la tónica, medio con la dominante, medio con la tónica y uno con la dominante.

I-IV-V-I: genera un compás con la tónica, uno con la subdominante, uno con la dominante y uno con la tónica.

I-II-V-I: genera un compás con la tónica, uno con la supertónica, uno con la dominante y uno con la tónica.



Resultado

```
K:C
E:Mayor
M:2/4
Q:120
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
[C8E8G8]|[C8E8G8]|[G8B8d8]|[G8B8d8]|
```

El resto de opciones genera el fragmento de duración total la suma de las duraciones de la progresión en semicorcheas. Dependiendo del compás podrán ser uno o varios compases. Si no ocupa uno o varios compases completos, el compás se rellena con silencios.

- **Generar Ritmo:** genera el ritmo seleccionado en el que las duraciones son hacer referencia a semicorcheas. Las notas serán la tónica de la tonalidad cuando se usa la N y silencios si se usa la Z.

Resultado

```
C3z2C3-|Cz7|zzzzzzzz|
```

Podemos aplicar un patrón rítmico a un grupo de notas (no acordes). Seleccionamos las notas y a continuación elegimos un patrón rítmico.

Resultado

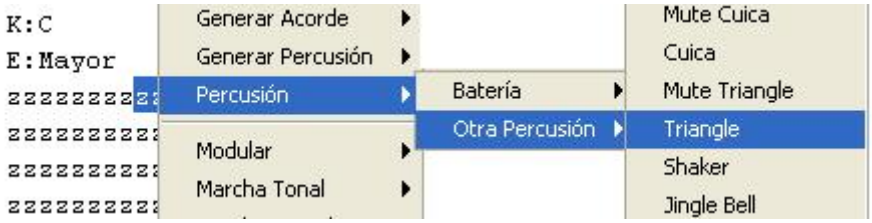
```
M:3/4
K:C
E:Mayor
D3EF4G4|D3EF4G4|zzzzzzzzzzz|zzzzzzzzzzz|
```

- **Generar Percusión:** genera una percusión. Es similar a generar Ritmo pero en este caso está especialmente indicado para las pistas 17 y 18 en las que cada nota midi tiene asociada un sonido de percusión. En esta ocasión no se utiliza la tónica sino la nota asociada al número indicado en la expresión rítmica, que en las pistas 17 y 18 están asociadas a determinados sonidos de percusión.

Resultado

```
M:2/4
^F,,,F,,,F,,,2z2F,,,F,,,|^F,,,4F,,,4|zzzzzzzz|
```

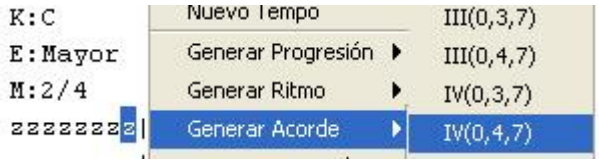
- **Percusión:** genera un sonido de percusión.



Resultado

```
K:C
E:Mayor
zzzzzzzzazzz|
```

- **Generar Acorde:** genera el acorde seleccionado. Todos los acordes se generan con duración 1. Hay que tener en cuenta que al generar el acorde no se eliminan notas del compás por lo tanto hay que controlar la duración del compás tras generar el acorde para no se exceda la duración permitida para el compás.



Resultado

```
zzzzzzz[FAc]z|zzzzzzz
```

3.2.8.Opciones de modificación

- **Modular Tonalidad:** modula el fragmento seleccionado a la tonalidad indicada. Deben seleccionarse uno o más compases enteros y la selección no debe terminar con el carácter de separación de compás '|'.

Resultado

```
[C4E4G4]GBdG|CEGC[C4E4G4]|
K:E
[E4G4B4]BdfB|EGBE[E4G4B4]|
```

- **Modular Escala:** modula el fragmento seleccionado a la escala seleccionada. Deben seleccionarse uno o más compases enteros y la selección no debe terminar con el carácter de separación de compás '|'.

Resultado

```
E:MixtaII
K:C
M:2/4
Q:120
CDEFG_A_Bc|zzzzzzzz|zzzzzzzz|zzzzzz:
zzzzzzzz|zzzzzzzz|zzzzzzzz|zzzzzzzz
```

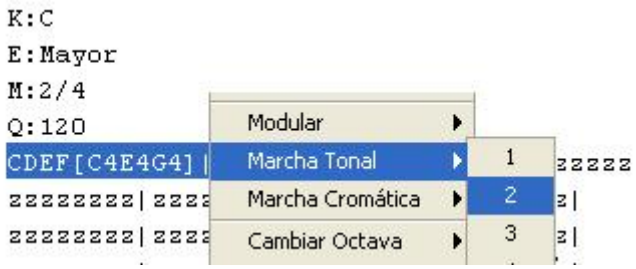
- **Marcha Cromática:** repite el fragmento seleccionado, tantas veces como se haya indicado. En cada repetición el fragmento se transporta un semitono más alto si el parámetro es positivo y un semitono más bajo si el parámetro es negativo.

Resultado

```
[E8G8c8]|F2A2c2F2|[F8_A8^c8]|^F2_B2^c2F2|
[^F8A8d8]|G2B2d2G2|zzzzzzzz|zzzzzzzz|
```

- **Marcha Tonal:** repite el fragmento seleccionado, tantas veces como se haya indicado. En cada repetición el fragmento se transporta una segunda más alta si el parámetro es positivo y una segunda más baja si

el parámetro es negativo. Cuando la nota a transportar no se encuentra en la escala siempre transportamos una segunda mayor.



Resultado

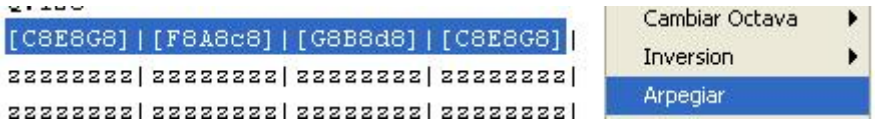
```
K:C
E:Mayor
M:2/4
Q:120
CDEF[C4E4G4]|DEFG[D4F4A4]|EFGA[E4G4B4]|
```

- **Inversión:** invierte el acorde seleccionado situando en el bajo la nota seleccionada en el parámetro.

Resultado

```
[E8G8c8]
```

- **Arpegiar:** arpegia el fragmento seleccionado. Se pueden arpegiar varios compases.



Resultado

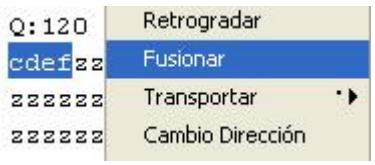
```
C2E2G2C2|F2A2c2F2|G2B2d2G2|C2E2G2C2|
```

- **Retrogradar:** invierte en el tiempo el orden del fragmento seleccionado.

Resultado

```
C2G2E2C2|G2d2B2G2|F2c2A2F2|C2G2E2C2|
[C8E8G8]|[G8B8d8]|[F8A8c8]|[C8E8G8]|
```

- **Fusionar:** fusiona en un acorde las notas seleccionadas. La duración del acorde se obtiene sumando las duraciones de las notas seleccionadas.



Resultado

```
Q:120
[c4d4e4f4]zzzz|zzzzzzzz
```

- **Transportar:** transporta el fragmento seleccionado el número de semitonos elegido.

Resultado

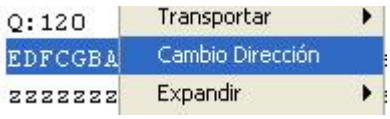
```
Q:120
D2E2[D4^F4A4]|zzzzzzzz
```

- **Expandir:** añade la nota que se encuentra a la distancia del intervalo seleccionado, a las notas seleccionadas que sean de la escala.

Resultado

```
Q:120
[C2E2][D2F2][C4E4G4B4]|zzzzzzzz
```

- **Cambio Dirección:** Tomando como referencia la primera nota realiza la reflexión de los intervalos que forman las siguientes notas con la nota referencia, a modo de imagen especular.

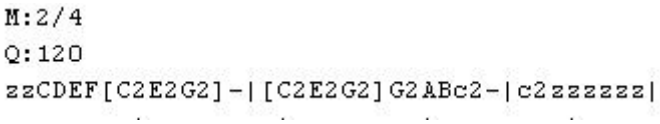


Resultado

```
Q:120
E^F_E_A^CA,B,z|zzzzzzzz
```

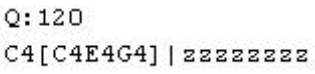
- **Trasladar:** traslada en el tiempo el fragmento seleccionado. Se desplaza en unidades de semicorcheas.

Resultado



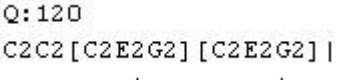
- **Aumentar X:** multiplica por 2 o por 3 las duraciones de las notas de los compases seleccionados.

Resultado



- **Dividir /:** divide entre 2 o entre 3 el fragmento seleccionado.

Resultado



- **Octavar:** añade la octava al fragmento seleccionado y divide la duración original, quedando la original con la mitad y la nueva a distancia de octava con la otra mitad.

Resultado

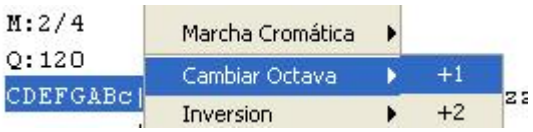
```
Q:120
C2c2G2g2|zzzzzzzz
```

- **Disminuir-Repetir:** es similar al típico trino pero permitimos que sea de más de dos notas.

Resultado

```
Q:120
C/D/E/C/D/E/C/D/E/C/D/E/C/D/E/C/|
```

- **Cambiar Octava:** cambia la octava del fragmento seleccionado. El parámetro indica cuántas octavas se suman o restan.



Resultado

```
Q:120
cdefgabc'|zzzzzzzz|
```

3.2.9.Uso de la Pianola

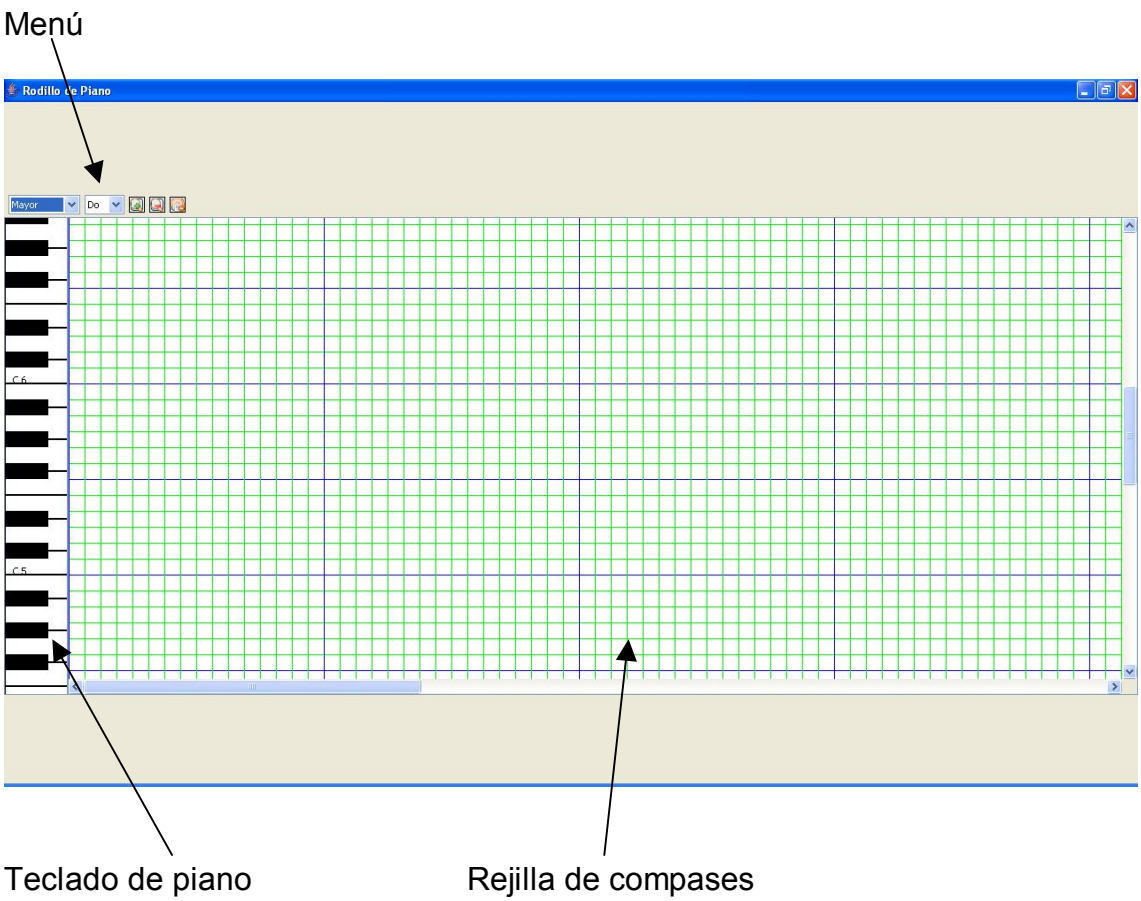
La Pianola aporta un modo diferente al formato ABC de editar y añadir notas musicales.
El protocolo midi es un estándar de comunicación entre instrumentos musicales electrónicos y/o computadores utilizado desde el principio de los años 80. La existencia de este tipo de herramienta en los secuenciadotes MIDI es una práctica habitual. En concreto, el desarrollo de la misma se ha inspirado en la aplicación "cakewalk".
La Pianola ofrece al usuario una interfaz en donde las notas musicales se representan con las 128 notas MIDI.

Mediante el uso de la Pianola se posibilita una edición y corrección de notas lo mas cómoda posible. El modo texto desaparece y la totalidad del uso de la pianola se hace con el ratón. De esta manera se puede crear una nota pinchando el botón izquierdo, alargar o acortar su duración con el botón derecho o cambiar una nota de posición arrastrándola con el ratón.

La pianola aporta una manera muy intuitiva de visualizar la partitura. Cada fila representa una nota diferente y la longitud del rectángulo marcado en esa fila su duración. Los cambios de compás se detectan visualmente por el distinto color con el que están dibujadas las líneas verticales. En todo momento se visualiza un teclado de piano puesto de forma transversal en el que cada tecla tiene asociada una fila.

Las líneas de distinto color y el teclado hacen posible ver gráficamente de una manera global la relación de las notas con respecto al tiempo. La Pianola no sólo aporta información visual, ya que la inserción y movimiento de notas se ven siempre acompañados del sonido real de la nota que se quiere añadir a la composición.

Apariencia general :



3.2.9.1.Menú de la Pianola

El rodillo de piano dispone de un pequeño menú a base de botones y menús desplegables.

- **botón de ampliación del tamaño de la rejilla de compases:** Cada vez que se presione este botón la resolución de la pianola se verá cambiada por una mayor. Tiene un valor límite en el cual el botón ya no tiene efecto. Este valor umbral se ha elegido en base a la idea de que hay un momento en que una resolución mayor impide un correcto uso de las funcionalidades de la pianola.

- **botón de disminución del tamaño de la rejilla de compases:** Del mismo modo que en el botón anterior, cada vez que se pincha en

este botón la resolución del rodillo de piano se ve modificado por una resolución menor. Dispondrá también de un cierto límite de actuación en el cual ya no se permitirá seguir disminuyendo el tamaño considerando que una resolución menor no permite un trabajo óptimo.

Tanto el botón de ampliación y disminución de tamaño tienen consecuencias respecto del eje X y del eje Y. De este modo la pianola crecerá o disminuirá de manera conjunta en anchura y longitud.

- botón de borrado:
  Este botón permite la eliminación de notas de la pianola. Una vez que se ha pulsado tendrá efecto hasta que se deshabilite. Si el botón se encuentra habilitado tendrá un color diferente de modo indicativo de que si se pincha sobre cualquier nota introducida con anterioridad se borrará de la partitura. Una vez se hayan borrado las notas deseadas se debe volver a pulsar recuperando su color original. Si se habilita el botón de borrado y se pincha en algún lugar del rodillo donde no se encuentra ninguna nota esta acción no tendrá ninguna consecuencia.

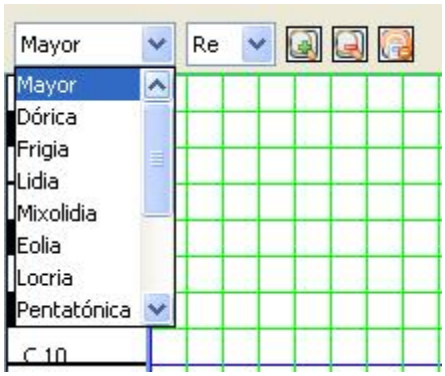
  Para insertar notas, este botón no puede estar seleccionado.

También dispone de un menú desplegable de selección de escala:

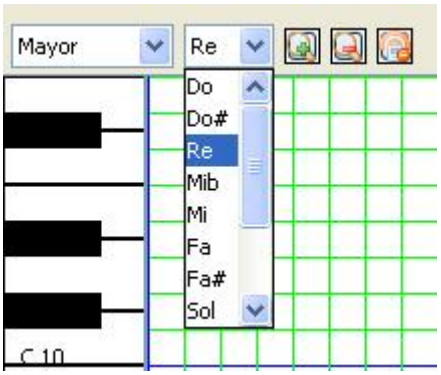
- menú desplegable de elección de escala:
  Permite seleccionar una de las siguientes escalas musicales:

  -Mayor
  -Dórica
  -Frigia
  -Lidia
  -Mixolidida
  -Eolia
  -Locria
  -Pentatónica
  -Mixta I
  -Mixta II
  -Menor Natural
  -Armónica
  -Melódica

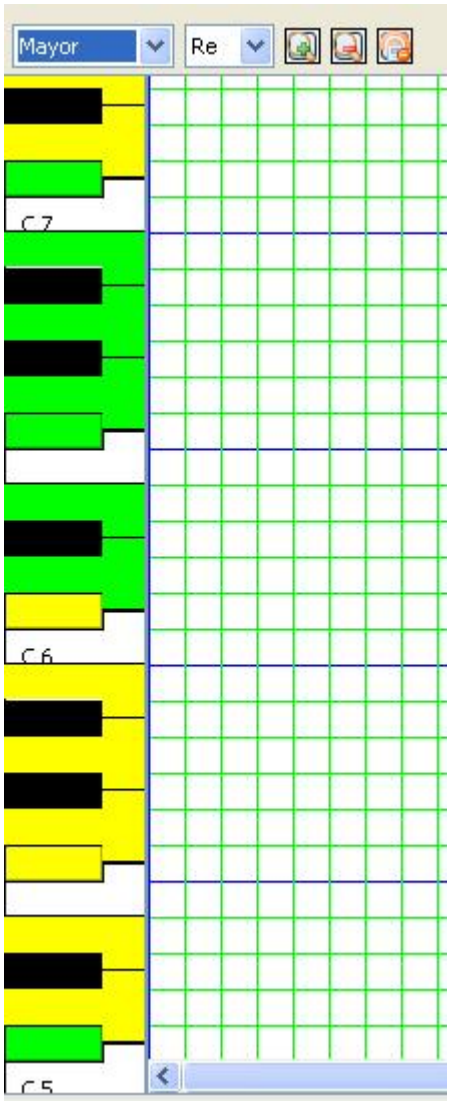
Una vez seleccionada, las notas de la escala seleccionada quedan dibujadas de distinto color en el teclado de piano transversal.



- **menú desplegable de elección de tonalidad:** determina la tonalidad que usará la escala seleccionada.



El par (escala,tonalidad) seleccionado determina las teclas que se pintarán de distinto color para distinguirlas del resto de teclas. Se alternan los colores dependiendo de la octava en la que nos encontremos para facilitar la visión



3.2.9.2.Teclado de Piano

Este componente se ha introducido como referencia visual en aras de facilitar el proceso de creación musical. Crea un paralelismo con el entorno real del músico. Cada vez que se introduce o modifica una nota en la rejilla, se tendrá en todo momento la referencia a la nota en un piano real. Cuenta pues de 128 teclas: una para cada sonido midi cubriendo un total de 11 octavas numeradas de 0 a 10.

Las correspondencias son las siguientes:

Octava	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Se ha introducido en cada tecla Do del piano el número de octava a la que hace referencia a modo de leyenda. De esto modo C0 representará el Do de la octava número 0, C1 el Do de la octava número 1 y así sucesivamente hasta el C10. Es una manera rápida y cómoda de saber en todo momento con la octava que se está trabajando. Aunque en todo momento se puede escuchar el sonido midi de la correspondiente nota del piano pulsando el botón izquierdo del ratón se ha preferido añadir esta leyenda para tener una referencia gráfica y no solo de sonido.

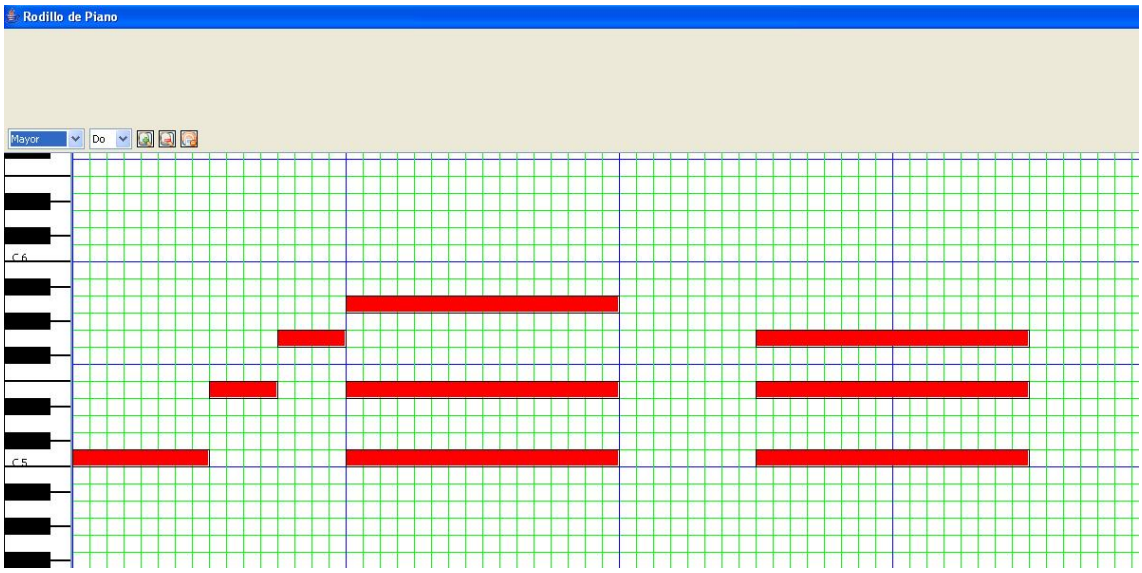
El teclado de piano da el soporte suficiente para mostrar la escala seleccionada por el usuario. De esta manera el compositor podrá contar con el apoyo gráfico de la escala seleccionada replicada por todas las octavas del piano. Para poder visualizar la totalidad del piano, el usuario dispone de un scroll vertical que le permitirá desplazarse por todas las teclas.

3.2.9.3.Rejilla de compases

Este componente es el elemento mas importante del rodillo de piano ya que es la herramienta a través de la cual se interactúa con la partitura que se va generando. Va a permitir al usuario añadir una nueva nota, mover una nota de lugar, borrar las notas deseadas, o cambiar la duración de la nota que se quiera así como una apariencia general de lo que se ha compuesto hasta el momento. Es por ello el componente central de la pianola que actúa a modo de “interfaz editable” de la partitura.

Orfeo genera automáticamente notación ABC a partir del contenido de la pianola. El panel de edición también muestra en notación ABC el contenido musical que se haya creado o modificado con la pianola. La pianola está diseñada para introducir fragmentos melódicos pero también permite introducir acordes.

A continuación se muestra la traducción a notación ABC que genera Orfeo a partir del contenido de la pianola de la siguiente captura.



C4_E2G2|A8|zzzz[C4_E4G4]-|[C4_E4G4]zzzz|

El ejemplo está hecho usando compás de 2/4. El valor de cada cuadro es de fusa.

La apariencia visual es una serie de líneas verdes horizontales que van marcando las diferentes notas midi yendo en todo momento en consonancia con el teclado de piano transversal. También dispone de una serie de líneas azules verticales que limitan la duración del compás y una serie de líneas verdes verticales que marcan la unidad de duración de la nota.

La unidad de medida de referencia (en el caso de Orfeo la fusa) se mostrará de manera gráfica por un cuadrado en donde su índice con respecto al eje de las ordenadas es directamente la nota midi que representa. De esta manera, (tomando como centro de los ejes X e Y la esquina superior izquierda de la rejilla), los cuadrados de la fila 0 tendrán un sonido midi 0, los cuadrados de la

fila 1 tendrán un sonido midi 1, hasta llegar a la fila 127 donde los cuadrados tienen un sonido midi 127. Como ya se ha dicho con anterioridad la unidad de duración con respecto al tiempo de estos cuadrados dependerá de la correspondiente duración del compás.

En relación a la modificación de la duración de la nota, el tamaño de la nota representado por la longitud del rectángulo coloreado en rojo se verá aumentado en una unidad musical más o por el contrario disminuido en una unidad musical menos dependiendo de la naturaleza de la modificación, Aportando en todo momento claridad visual del tamaño de cada nota en la partitura. Si seleccionamos medio cuadro corresponde a la duración de una fusa.

Para poder seleccionar la nota que se desee (entre el rango 0-127), el usuario puede desplazarse por la rejilla gracias a un scroll vertical. Para facilitar su uso, si el usuario está moviendo el puntero del ratón para añadir una nota o cambiarla de lugar y la pianola detecta que el puntero se sale del rango superior o inferior representado en ese momento en la rejilla, el movimiento pertinente del scroll se hará de modo automático.

El scroll horizontal se comporta de un modo similar al scroll vertical, haciendo auto-scroll en el momento en que la nota se encuentra en los límites horizontales de la ventana actual de representación gráfica. El número de filas de la rejilla es un número finito y conocido en concordancia con el número de notas midi existentes. Por el contrario el número de columnas depende de la duración de cada partitura personal del compositor. ORFEO no pondrá límite a este parámetro. En el momento en que la composición va creciendo, el número de columnas lo hace de un modo automático a la vez no queriendo restringir la capacidad creadora del artista. Como toda aplicación informática el límite de ORFEO estará marcado por las características y recursos de la máquina física que se está utilizando.

3.2.9.4. Funcionalidad de la Pianola

a)Inserción de nota:

La manera de añadir una nota en la rejilla de compases se hace de una modo fácil y natural. Se debe pinchar el botón izquierdo en el lugar de la rejilla donde se quiere que empiece a sonar y desplazar el ratón sin soltar el botón en función de la duración que se le quiera dar. No podrán quedar notas solapadas en ningún momento. Solo se permitirá que dos notas puedan sonar en un mismo tiempo si empiezan a la vez. Dando la posibilidad de crear acordes. Pero si dos notas empiezan en lugares diferentes y en algún momento su sonido se generara de manera simultánea el proceso de inserción se ve cancelado y no se permite tal suceso.

b)Movimiento de nota:

Se permite el movimiento de notas ya creadas a lo largo de la rejilla. Para llevar a cabo esta tarea, simplemente se debe pinchar con el botón izquierdo del ratón en la nota que se desea cambiar de sitio y hacer el desplazamiento deseado sin soltar el ratón. El nuevo emplazamiento de la nota se verá confirmado al soltar el botón izquierdo del ratón que hasta este momento estaba pulsado. Al igual que en el proceso de inserción la rejilla permite realizar el movimiento en función de las restricciones establecidas con anterioridad. Si al establecer el nuevo emplazamiento de la nueva nota, la rejilla detecta que en algún momento la nota sonará a la vez que alguna de las notas insertadas con anterioridad y no empezando en el mismo lugar que esta última, el proceso de desplazamiento se verá cancelado llevando la nota a su lugar de origen. Si por el contrario la nota, una vez cambiada de lugar, quedara en un punto de empiece igual que el de cualquier otra nota añadida con anterioridad se estaría creando un nuevo acorde.

c)Cambio de duración:

Este proceso se lleva a cabo con la utilización del botón derecho del ratón. Si se desea cambiar la duración de una nota ya introducida a la partitura, en vez de proceder a su borrado y a una nueva creación con la duración deseada, la rejilla aporta esta funcionalidad agilizando las tareas de edición. Se debe pulsar el botón derecho en la nota que se desea modificar de tamaño y hacer un desplazamiento hacia la derecha si se desea aumentar la duración o un desplazamiento hacia la izquierda si lo que se desea es disminuir la duración, (los desplazamientos se efectúan sin soltar el botón). Una vez efectuada la modificación, al soltar el botón derecho los nuevos cambios quedarán establecidos.

*La pianola no puede mostrar dosillos ni tresillos. Tampoco permite dibujar notas que comenzando en posiciones distintas del eje x compartan espacio en el tiempo. Esta restricción es producto de que Orfeo no permite trabajar con varias voces independientes en una misma pista, pero sí con acordes.

4.¿Cómo está hecho ORFEO?

En el desarrollo de la aplicación se han empleado dos lenguajes de programación: java y prolog.

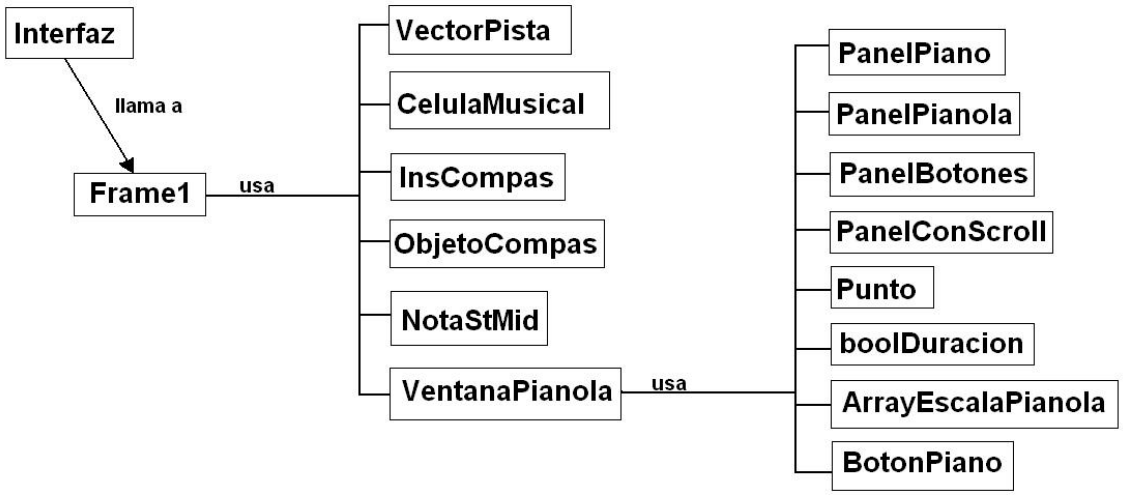
Java ha sido utilizado para crear la interfaz y la pianola, mientras que Prolog se ha usado para desarrollar las reglas de creación y modificación de fragmentos musicales.

Se almacena en vectores la información de cada pista: número de compases, tipo de compás, escala, tonalidad, tempo, volumen, instrumento midi asociado y el contenido musical de cada compás.

La pianola usa un vector que almacena las notas que deben pintarse en el rodillo de piano.

Las reglas escritas en prolog trabajan con listas.

El siguiente diagrama muestra las clases de la aplicación y sus relaciones:



4.1.Interfaz principal

Al estar realizada en Java se explican a continuación las estructuras de datos y las clases más importantes de la interfaz. Se ha hecho uso de las librerías AWT y Swing para crear los paneles, botones, combos y scrolls.

4.1.1.Estructuras de datos

Manejamos varias representaciones: lenguaje Orfeo alto nivel basado en ABC, representación a bajo nivel basándonos en la representación midi y representación en prolog.

Para que estas tres representaciones puedan coexistir en la aplicación tenemos varios traductores:

ParserABCProlog: Traduce un fragmento en notación ABC al formato que maneja prolog (explicado en el capítulo 4.3).

String ParserABCProlog (String in)

Ejemplo:

Entrada: G4Z4|[C8E8G8]

Salida: [(67,4),(1000,4),s, [(60,8), (64,8), (67,8)]]

ParserABCMidi: traduce el contenido del campo edición al formato en el que las notas se representan con los objetos CelulaMusical. Estas células se almacenan en vectores que a su vez se guardan en el elemento correspondiente a la pista seleccionada en el VectorComposicion.

boolean ParserABCMidi (String in)

Ejemplo:

G4Z4|[C8E8G8]

Crea un vector formado por dos vectores V1 y V2 que almacena objetos de la clase CelulaMusical con el siguiente contenido:

V1:
V1[0]
NotaSt=”67”
Duracion=8
NumeroNotas=1
Acorde=null

V1[1]
NotaSt=”1000”
Duracion=8
NumeroNotas=1
Acorde=null

V2:
V2[0] NotaSt=null
Duracion
NumeroNotas=3
Acorde= Vector Vacorde con objetos CelulaMusical

Vacorde[0]
NotaSt=”60”
Duracion=16
NumeroNotas=1
Acorde=null
Vacorde[1]
NotaSt=”64”
Duracion=16
NumeroNotas=1
Acorde=null
Vacorde[2]
NotaSt=”67”
Duracion=16
NumeroNotas=1
Acorde=null

ParserMidiABC: traduce el contenido de una pista del VectorComposicion (indicado con el parámetro *lapista*) al formato que se muestra en el campo de edición. El parámetro *compases* indica cada cuantos compases se debe introducir un salto de línea.

String ParserMidiABC(int lapista, int compases)

ParserOutMidiABC: traduce el contenido de una pista del vector *VectorComposicion* (indicado con el parámetro *lapista*) al formato ABC. El parámetro *compases* indica cada cuantos compases se debe introducir un salto de línea. Se utiliza para crear archivos en formato ABC para reproducir, mostrar partitura, exportar ABC, exportarMIDI y exportar PS.

String ParserOutMidiABC(int lapista, int compases)

ParserPrologABC: Traduce un fragmento musical en formato prolog al formato empleado en el campo de edición.

String ParserPrologABC(String in)

Las estructuras de datos más importantes en la parte de la interfaz son:

- *vector VectorComposicion:* Contiene la información de cada pista. Almacena elementos del tipo *VectorPista*.
- *Vector VProgresiones*: almacena las progresiones armónicas que se cargan por defecto y las que se cargan desde archivo. Almacena strings.
- *Vector VRitmos*: almacena las patrones rítmicos que se cargan por defecto y las que se cargan desde archivo. Almacena strings.
- *Vector VPercusiones*: almacena las percusiones que se cargan por defecto y las que se cargan desde archivo. Almacena strings.
- *Vector VAcordes*: almacena los acordes que se cargan por defecto y las que se cargan desde archivo. Almacena strings.
- *VentanaPianola VPianola*: el objeto encargado de crear la pianola.

A continuación se citan los cuatro métodos relacionados con la comunicación entre la interfaz y la pianola y la creación de esta última.

PianolaToMidi: Este método es el encargado de traducir la información de la pianola al formato de los objetos CelulaMusical.

void PianolaToMidi()

VerPianola: se encarga de crear un objeto VentanaPianola para crear y mostrar la pianola.

void VerPianola()

ActualizaRodilloPianola: actualiza la pianola con la información del campo de edición.

void ActualizaRodilloPianola()

La interfaz principal se comunica son la pianola compartiendo un vector llamado *ElVector*, que almacena objetos de la clase Punto, que representan las notas que se pintan en la pianola.

Las variables booleanas *hayPianola* y *PianolaActiva* se encargan de definir el estado de trabajo y de esta manera saber si hay pianola y está activa o es el campo de edición es el que está activo.

Posibilidades:

	True
hayPianola	x
PianolaActiva	x

En este estado el control recae sobre la pianola y los cambios que se hagan el rodillo de piano son guardados y se reflejan posteriormente en el campo de edición de la interfaz principal.

	True	False
hayPianola	x	
PianolaActiva		x

En este estado el control recae en la interfaz principal y los cambios que se realizan en el campo edición se reflejan posteriormente en la pianola.

	False
hayPianola	x
PianolaActiva	_

En este estado el control recae en la interfaz principal y los cambios que se realizan en el campo edición se reflejan posteriormente en la pianola si es visualizada puesto que cuando la variable *hayPianola* tiene el valor false significa la pianola no está siendo visualizada.

Tanto los cambios producidos en el campo de edición que se ven reflejados en la pianola, como los producidos por la pianola que se ven reflejados en el campo de edición se producen sin que el usuario tenga que realizar ninguna acción especial. La aplicación detecta cuál es la zona activa de trabajo y realiza las actualizaciones oportunas.

4.1.2.Clases

- **Interfaz**: es la clase que alberga el método main(), desde el se inicia la aplicación creando e invocando un objeto de la clase *Frame1*.

- **Frame1**: es la clase que crea la interfaz gráfica, comunica la interfaz con la pianola y la interfaz con el código prolog.

- **VectorPista:** estos objetos almacenan la información de cada pista.

  int instrumento: entero que identifica el instrumento midi con el que se reproduce la pista.

  Vector VectorMusica: vector que contiene tantos vectores como compases tenga la pista. Cada uno de estos vectores almacena objetos *CelulaMusical.*

- **CelulaMusical:** los objetos de esta clase representan las instrucciones y las notas.

  String[] NotaSt : númeromidi que identifica la nota.*

  int Duracion: duración de la nota en fusas.

  int NumeroNotas:

  Si NumeroNotas == 0 Se trata de una ligadura **
  Si NumeroNotas == 1 Nota individual
  Si NumeroNotas > 1 Acorde
  Si NumeroNotas == -2 Dosillo
  Si NumeroNotas == -3 Tresillo
  Si NumeroNotas == -10 Instrucción ***

  Vector Acorde: Vector de objetos CelulaMusical. Está vacío si NumeroNotas==0 ó ==1 . En otro caso contiene las notas que forman el acorde (si NumeroNotas >1) o el dosillo(NumerNotas=-2) o tresillo(NumeroNotas=-3).

***** En el caso de los silencios, al no tener asociada una nota midi indicamos que es un silencio con el número 1000.
****** El campo NotaSt contendrá el valor 1001 para indicar que es una ligadura.
*******En este caso el campo NotaSt contiene la instrucción de Tonalidad, escala, compás o tempo.

- **InsCompas:** los objetos de esta clase almacenan las instrucciones y el compás en que aparecen para recuperar las información de tonalidad, escala, compás y tempo al pasar de la pianola al campo de edición.

  String Instruccion: guarda las instrucción

  int NumCompas: guarda el compás en el que aparece la instrucción

- **NotaStMid:** sirve para mostrar correctamente las notas en formato ABC al traducirlas desde el formato MIDI.

  String NotaSt: guarda la nota en formato ABC.

  int NotaMid: guarda el número midi asociado a la nota.

- **ObjetoCompas:** estos objetos se almacenan en el vector *VectorCambiosCompas* para que la pianola sepa dónde pintar las divisorias de compás.

  int CompasInicio: guarda dónde comienza el tipo de compás

  int CompasFin: guarda dónde termina el tipo de compás

  int Duracion: guarda la duración del compás

4.2.Pianola

Para realizar la implementación de la interfaz de usuario de la pianola se ha utilizado tanto AWT como Swing. Debido a que el lenguaje de programación Java es independiente de la plataforma en que se ejecuten sus aplicaciones, tanto el AWT como el Swing es independiente de la plataforma en que se ejecute. Proporcionan un conjunto de herramientas para la construcción de interfaces gráficas que tienen apariencia y comportamiento semejante en todas las plataformas que se ejecute. Fueron diseñados pensando en que el programador no tuviese que preocuparse de detalles como controlar el movimiento del ratón o leer el teclado. Constituyen unas librerías de clases orientadas a objeto para cubrir estos recursos y servicios de bajo nivel.

Básicamente la clase VentanaPianola está constituida por tres paneles principales:

-El primer panel “PanelBotones” se utiliza como contenedor de los botones y del menú desplegable de escalas del menú principal del rodillo de piano. Para ello se ha creado una nueva clase que extiende de “JPanel” donde se ha sobrescrito el método “getPreferredSize()” para poder modificar su tamaño.

-El segundo panel es “PanelScroll2” que es un JScrollPane con el scrollbar horizontal deshabilitado. En su interior encontramos un “PanelPiano” que extiende “JPanel” y reescribe el método “paintComponent” para pintar el teclado del piano con la ayuda de la clase Graphics. Su layout es un

BorderLayout.WEST para que quede localizado a la izquierda de “PanelScroll” que será el panel principal.

-El objeto “PanelScroll” sirve para pintar la rejilla de la pianola. Para ello se ha creado una clase “PanelConScroll” que extiende “JScrollPane” y sobrescribe “paintComponent” para que refresque su contenido. Como la longitud de la rejilla puede crecer dinámicamente de manera indefinida hacia la derecha se habilita un scrollbar horizontal. El contenido de “PanelScroll” no es mas que un objeto de la clase “PanelPianola”. Esta última clase extiende “JPanel” y sobrescribe el método “paintComponent” para pintar correctamente todas las líneas horizontales y verticales de la rejilla y las notas introducidas por el usuario de la aplicación.

El scrollbar vertical que permite al usuario moverse de arriba a abajo para seleccionar la nota deseada tiene que ser el mismo para los dos últimos paneles. Ya que el movimiento vertical realizado en cualquiera de estos paneles tiene que ir acompasado.

Tanto **PanelPianola** como **PanelPiano** son clases parametrizadas en función de un valor “tamCuadro” que almacena el tamaño del cuadrado de la rejilla. De este modo si el usuario pincha en algún botón de modificación de tamaño de la resolución de la pianola, basta con cambiar dinámicamente el valor del parámetro y refrescar los paneles.

La estructura de datos para representar una nota es la clase **Punto**. Tiene cuatro atributos privados:

x: es un entero y almacena la fila de la nota con respecto a la rejilla tomando el centro de las coordenadas cartesianas la esquina superior izquierda.
Como cada fila representa una nota midi diferente, es fácil hacer corresponder la fila con la nota midi que representa.

y: guarda la información de la columna dónde empieza la nota.

dura: es la duración en número de cuadrados de la nota.

temp: es un valor booleano que informa si la nota es temporal. Como una de las funcionalidades básicas del rodillo es permitir el movimiento de una nota arrastrando el ratón, este atributo nos dice que esta nota todavía no tiene un lugar definitivo. Al cambiar la posición de una nota se crean muchas notas temporales que deben ser pintadas pero que una vez fijada la nueva posición deben desaparecer.

ligado: es una variable booleana que sirve para detectar si la nota que representa el punto está ligada. Almacena el valor *true* cuando está ligada y *false* si no está ligada.

Para procurar un fácil manejo del rodillo de piano se ha implementado un scroll automático cuando se detecta que una nota intenta rebasar los bordes de la parte visible de la rejilla. Para ello es necesario hacer corresponder la posición global de la nota dentro de la rejilla con la posición que ocupa dentro de la parte visible. Ya que cualquier evento del ratón se localiza en función de la parte de la rejilla que en esos momentos es visible.

En el proceso de desarrollo, se pensó en conseguir una rejilla que pudiera crecer hacia la derecha de una manera “infinita” (limitado por supuesto por los recursos de la máquina física donde se ejecuta la aplicación). Se marcó el objetivo de que ORFEO no limitara el tamaño de las columnas de la rejilla de la pianola. Para ello, se creó un nuevo parámetro “numColum” en la clase “PanelPianola” para almacenar la información del número de columnas actuales. Una vez parametrizada la clase, se añadieron nuevas sentencias de control en la parte donde se implementa el auto-scroll para detectar que se alcanzaba el final de la rejilla. Si se produce tal suceso, basta con actualizar el parámetro “numColum” por un valor mayor y refrescar el contenido del panel. El efecto conseguido es una rejilla que aumenta de tamaño de manera automática y dinámica según las necesidades del usuario.

4.3.Orfeo Prolog

4.3.1.Entorno de Trabajo

Para realizar la parte de nuestra aplicación que comprende los predicados lógicos que tratan las notas y aplican diversas transformaciones musicales a nuestra composición se precisaba un entorno capaz de tratar algoritmos con la mayor fluidez posible. Para lograrlo se decidió utilizar el lenguaje Prolog, y en concreto el entorno de desarrollo de Prolog denominado SWI-Prolog.

Este entorno de trabajo es de libre distribución, como el resto de las tecnologías y programas que se han utilizado en la aplicación. Para descargar la versión que se precise en función del sistema operativo, obtener información sobre el modo en que trabaja el entorno, o consultar manuales y posibles actualizaciones, basta con consultar la página web de SWI-Prolog: http://www.swi-prolog.org/

Por qué usar SWI-Prolog:

Se optó por usar este entorno de trabajo, además de porque siendo de libre distribución se disponía de mucha más facilidad para hallar cualquier tipo de información que se precisara, por su gran velocidad de desarrollo, que ha facilitado el uso de predicados que retrasarían un programa escrito en otros lenguajes, así como por ser un entorno ampliamente conocido y utilizado en múltiples áreas de desarrollo e investigación. Además el hecho de que sea una aplicación ligera simplifica su manejo e instalación.

SWI-Prolog es un entorno con grandes ventajas para una aplicación como la nuestra, que trata con lenguajes formales. Permite de manera rápida implementar las transformaciones musicales propias de nuestra aplicación, que son tratadas como transformaciones algebraicas por el entorno. Sin embargo, para el aspecto gráfico del proyecto se precisa contar con un entorno que tenga unas opciones gráficas más desarrolladas que las que ofrece este sistema, para lograr una facilidad de uso mayor y hacerlo más atrayente a la vista. Por este motivo la aplicación comprende el uso de la tecnología Java, y por ello se precisa un sistema para comunicar ambos entornos.

Además, el hecho de que existan versiones para Mac y Linux y que el código sea altamente portable sería una gran ayuda en caso de que quisiéramos que nuestra aplicación fuera multiplataforma en un futuro frente a otros programas sin estas facilidades.

4.3.2.Trabajando con Java: TUPROLOG

Para poder comunicar los dos entornos de programación de nuestro proyecto se necesita un parser que traduzca nuestro código y actúe de enlace entre ambas tecnologías. Este enlace es tuProlog.

4.3.2.1.¿Qué es tuProlog?:

TuProlog es un motor de Prolog basado en java y diseñado para aplicaciones de Internet e infraestructurales. Por esta razón, tuProlog está diseñado para destacar en algunos aspectos que le hacen interesante para nuestro tipo de aplicación en concreto:

Tiene requerimientos mínimos para poder ser ejecutado, ya que solamente requiere la presencia de la máquina virtual de Java y una invocación sobre un archivo .jar simple, y su núcleo es mínimo, toma la forma de un pequeño objeto Java que contiene sólo las propiedades más esenciales de un motor de Prolog.

TuProlog está caracterizado por un tener un minúsculo aunque eficiente motor de Prolog, fácilmente utilizable como un objeto autónomo de Java, accesible por medio de un interfaz mínimo. Su minimalismo le hace conveniente para construir applets, agentes de Internet e infraestructuras de poco peso, donde el grosor de software y la sobrecarga pueden ser simplemente inaceptables.

Además es altamente configurable, gracias a la carga y descarga de predicados, functores y operadores integrados en bibliotecas. Todo ello posibilita que a integración entre Prolog y Java sea tan profunda y limpia como es posible.

Dónde encontrarlo:

Para poder utilizar tuProlog sólo tenemos que acceder a su página web y descargarlo: http://www.alice.unibo.it/tuProlog/

Allí podemos obtener información completa sobre sus ampliaciones, librerías, guías de uso y demás contenidos que nos han resultado de gran ayuda a lo largo del desarrollo de la aplicación.

4.3.3.Sintaxis utilizada y Estructuras de Datos

En este apartado vamos a ver con detenimiento la sintaxis que se utiliza en Prolog así como las principales estructuras de datos que han sido utilizadas en la aplicación Prolog, que serán en la mayoría de los casos listas de objetos, dada la naturaleza de Prolog

La Estructura Principal = Lista de Notas

Las listas de notas Midi serán las estructuras principales del programa, sobre ellas se aplicarán las reglas y transformaciones en función de otros parámetros que serán normalmente enteros o bien otras listas.

Para usar con Prolog una lista de datos que represente una melodía se hará de la manera siguiente:

[(60, 4), (64, 4), 's', [(60,8), (67, 8)], 's']

⇨Que en notación ABC sería: C4E4|[C8G8]|

-Cada paréntesis formado por dos componentes numéricas, está representando una nota. Las componentes (X,Y) indican la nota que se está tratando (X) y la duración (Y) de la misma.

-Las notas pueden tomar valores numéricos comprendidos entre 0 y 127, representando así las 128 notas posibles en Midi.

-Las duraciones pueden ser en principio cualquier número natural positivo, siendo un tipo u otro de nota dependiendo de su duración.

Acordes:

Una lista de pares dentro de la lista que recibe Prolog es la notación que se ha dado a los acordes, está formada por pares que representan sus notas y su duración al igual que las notas sueltas. Un ejemplo de acorde lo tenemos en la lista anterior, es el formado por: [(60, 8), (67, 8)].

Marcas de fin de compás y ligadura:

En las listas que trata Prolog se usan también unas marcas de final de compás, que son necesarias para tratar por separado las notas y acordes de distintos compases Estas marcas consisten en la letra 's', entre comillas simples, este carácter lo reconocerán nuestros predicados como un carácter reservado y lo tratarán en consecuencia.

Además de marcas de final de compás, tenemos la marca de ligado entre dos o más compases. Esta marca se usará en el caso en que la duración de una nota no quepa en un compás (por haber más notas en el mismo compás y tener más duración que la restante en el compás o bien por tener una duración mayor a la del propio compás) y por tanto queden ligados dos o más compases. La marca consiste en usar un par (1001,0) seguido de la 's' usada para marcar el final de un compás normalmente. Se emplea un número que ninguna nota puede tomar (1001) y una duración que tampoco puede darse en ningún par ordinario (0). A continuación tenemos un ejemplo sencillo:

[(60, 4), (64, 4), (1001,0), 's', (64,2), (67, 6), 's']

Silencios:

Los silencios se representan en Prolog mediante un par del tipo: (1000, Duración) en la lista. Siendo '1000' el número especial reservado a tal efecto, y Duración la duración del silencio. En el siguiente ejemplo podemos ver una lista con una nota, un silencio con duración cuatro y un acorde:

[(60, 4), (1000, 4), ‘s’, [(60,8), (67, 8)], ‘s’]

Dosillos y Tresillos:

Los dosillos y tresillos son agrupaciones de dos o tres notas, su tratamiento requiere el uso de una sintaxis especial para distinguirlos de acordes y notas ordinarias. Para representar dosillos y tresillos se ha utilizado la siguiente notación:

-En el caso del dosillo: ['d',(Nota1, Dur1),(Nota2, Dur2),'f'], siendo 'd' la marca de inicio de dosillo, y 'f' la marca de finalización del mismo.

-En el caso del tresillo hay dos opciones, que tenga dos notas de distinta duración o tres notas de igual duración, pero siempre usa las dos mismas marcas:

-Con dos notas: ['t',(Nota1,Dur1),(Nota2,Dur2),'f']
-Con tres notas: ['t',(Nota1,Dur),(Grado2,Dur2), (Grado3,Dur3),'f'],

⇨Siendo 't' la marca de inicio de tresillo y 'f' de nuevo la marca del final.

-Además, los dosillos y tresillos pueden estar formados por acordes en vez de por notas sueltas, como en el ejemplo siguiente:

[['d',[(NotaMid, Dur), (NotaMid2,Dur), (NotaMid3,Dur)], [(NotaMid4,Dur), (NotaMid5,Dur)]'f '| R]]

⇨ Lista formada por un dosillo formado a su vez por dos acordes

Además de listas de notas formando melodías se usan también otras listas y parámetros que representan números enteros, pero detallaremos su notación y demás peculiaridades en la siguiente sección, en que trataremos la estructura general de la aplicación.

4.3.4.Estructura del Programa

En este apartado se va a comentar la funcionalidad y sintaxis de los predicados Prolog más importantes de la aplicación para tener una visión más cercana a la realidad del funcionamiento de la misma. Los predicados se encuentran divididos en predicados de creación de secuencias musicales y de modificación de las mismas. Estas secuencias musicales se corresponden en Prolog con listas de objetos que contendrán las notas de las composiciones. El primer grupo contendrá los predicados encargados de crear listas de notas

para formar nuestra composición en función a determinados parámetros y los segundos de aplicarles las transformaciones musicales que les dan nombre.

4.3.4.1.Reglas de Modificación de Secuencias Musicales:

Generar:

Generar es el predicado principal de la aplicación. Crea un determinado número de compases en función del compás, que devuelve formando una lista según las especificaciones que tengan sus parámetros de entrada. El aspecto que tendrá el predicado es el siguiente:
generar(Tonalidad, TipoEscala, Compas, Octava, ListaEntrada, ListaGenerada).

- Tonalidad: Número entero que indica la tonalidad que será aplicada a las notas que generaré el predicado. Puede tomar cualquier valor par comprendido entre el 0 y el 22.

- TipoEscala: Indica el tipo de escala que queremos para nuestros compases, es un natural. Puede tomar valores desde el cero al nueve y además el 20, 21 y 22.

- Compas: Define el tipo de compás que usamos, lo hemos notado usando dos cifras numéricas positivas con un guión entre ambas; siendo sus posibles valores: 2-4, 3-4, 4-4, 3-8, 6-8, 9-8 y12-8.

- Octava: Nos señala la octava en la que trabajamos mediante otro número natural, siendo cero su valor por defecto.

- La lista de entrada ListaEntrada tendrá elementos del tipo: [(Grado,Duración), (G,D),...], aunque también aceptará dosillos o tresillos, pero no acordes.

  - Grado puede ser= 1, 2, 3, 4, 5, 6 ó 7 se utilizará en el predicado auxiliar dameAcorde, que devuelve un acorde formado por tres notas en función grado de la lista inicial, así como de su Tonalidad y TipoEscala.

  -Duración= Número natural que indica la duración del par

- ListaGenerada: Lista de salida que contendrá una lista formada por pares de notas (NotaMidi, Duracion) siendo ambos parámetros números enteros, formando acordes de tres notas que son representados a su vez por una lista anidada de pares iguales a los descritos.

Funcionamiento: El predicado llama a un predicado auxiliar llamado generarAux, necesario porque incluye un nuevo parámetro: DuraciónAcumulada, que lleva la duración que hemos gastado del compás actual, es necesaria para poder saber en qué lugares se han de colocar las marcas de final de compás y ligadura. Este predicado, que realiza todo el trabajo, va tratando en cada iteración a una pareja de la lista de entrada,

formando un acorde de tres notas con cada una de ellas gracias al predicado auxiliar dameAcorde. Si al formar el acorde la duración del mismo más la que se lleve acumulada de acordes anteriores excede a la duración del Compás, se debe poner la correspondiente marca de final de compás. Si excede a la duración del compás se pondrá la marca de ligadura, y se seguirá tratando al mismo par con la duración que le reste en un nuevo compás. Si por el contrario la duración es menor que la que le queda al compás actual, se continuará con la siguiente pareja pero aumentando la duración acumulada. Además, si lo precisa la lista de entrada, se introducirán silencios para completar la duración de los compases.

⇨Ejemplo: generar(0, 0, 2-4, 5, [(1,12), (4,2), (5,2), (1,4)], LOut).

> LOut=['[',(60, 8),(64, 8),(67, 8),']','|','[', (60,4),(64, 4),(67, 4),']', '[', (65,2), (69,2),(72,2)',]','['(67,2),(71,2),(74,2) ']', '|',[', (60,4),(64, 4), (67, 4),']', (1000,4)].

-La duración de un grado puede exceder un compás. En el ejemplo (1,12) al sercompás 2-4(duración 8) ocupa 1 compás y medio: '[',(60, 8),(64, 8),(67, 8),']','|','[', (60,4),(64, 4),(67, 4),']',

-Se completa la duración si hace falta con un silencio ((1000,4) en el ejemplo). En el ejemplo ocurre en el tercer compás: (1,4): DuracionCompás – DuraciónAcumulada: 8 – 4 = 4, se rellena con un silencio de duración cuatro.

GenerarRitmo:

Este predicado es muy similar a generar, pero en vez de generar una lista formada por acordes, genera una lista de notas sueltas. Su sintaxis es la siguiente:

generarRitmo(Tonalidad, TipoEscala, Compas, Octava, ListaEntrada, ListaGenerada).

Como se puede apreciar, los parámetros son los mismos que usa generar, los predicados se distinguen por el modo en que construyen la salida. Mientras que generar llama a un predicado auxiliar que le devuelve un acorde, generarRitmo hace lo propio con uno que sólo genera una nota por cada par.

Sus parámetros son iguales, admiten las mismas entradas y se comportan de igual modo.

⇨Ejemplo: generarRitmo(0, 0, 2-4, 5, [(1,12), (4,2), (5,2), (1,4)], LOut).

LOut = [(60, 12), (65, 2), (67, 2), s, (60, 4), (1000, 12), s]

GenerarRitmoNotas:

Este predicado es similar a generarRitmo, pero usa dos listas de entrada con las que va construyendo la salida, su sintaxis es:

generaRitmoNotas(Compas, ListaIN1, ListaIN2, ListaOut).

Esta regla funciona leyendo notas de listaIN1, si la nota es distinta de 1000 se lee una nota de listaIn2 y se añade a la salida las nota de listaIN2 con la duración de listaIN1. Si la nota de listaIN1 es 1000 se añade el par directamente a la salida.

El predicado se finaliza cuando se acaban los elementos de listaIN1. Si hay más elementos en listaIN1 que en ListaIn2, los elementos que hay de más se añaden a la salida pero poniendo en la nota 1000.

⇨Ejemplo: generaRitmoNotas(2-4,[(1,8),(1,4),(1,4)],[(23,_),(25,_)],Lout).

Lout=[(23,8),(25,4),(1000,4),s]

GenerarPercusión:

Este nuevo predicado también es muy similar a los anteriores, dado que su finalidad también es crear una lista de notas que formará parte de nuestra composición. El aspecto que presenta es el siguiente:

generarPercusion(Compás, ListaEntrada, ListaSalida).

Como podemos observar, es una versión simplificada de los anteriores predicados. La diferencia radica en que en este caso simplemente se irán tratando los pares de la lista de entrada y añadiéndolos a la ListaSalida sin modificarlos, sólo añadiendo las marcas que sean precisas en cada caso.

⇨Ejemplo: generarPercusion(6-8,[(35,6),(1000,3),(45,4)], L).

L = [(35, 6), (1000, 3), (45, 3), (1001, 0), s, (45, 1), (1000, 11), s]

Tema:

Este predicado fue la primera versión de lo que ahora es Generar, la principal diferencia radica en que mediante el uso de Tema, la creación de acordes para la composición no acepta listas de entrada. El número de compases que se generan se indica con un parámetro de entrada. Su sintaxis es de la forma:

tema(Numero, Tonalidad, TipoEscala, NumCompases, Compas, Octava, ListaGenerada).

Los dos parámetros que difieren con generar (a parte de la ausencia de la lista de entrada) son:

Numero, que es un número entre el uno y el cinco cuya función es simplemente discriminar entre los distintos enunciados del predicado, de modo que llamará a unas u otras funciones auxiliares; y NumCompases que indica el número de compases que deben generarse.

4.3.4.2. Reglas de Modificación de Secuencias Musicales:

Inversión:

Esta primera regla de modificación de listas realiza la regla musical de inversión al acorde que le pasamos en la variable Acorde. Su sintaxis:

inversion(Inversión, Acorde, AcordeInv).

- Inversión: Es de tipo natural, siendo su tope el número de elementos que haya en el acorde. Indica el número de inversión que realizaremos.
- Acorde: Es un acorde formado por pares de números, como los ya vistos.
- AcordeInv: Es el parámetro de salida que contiene el acorde invertido.

Funcionamiento: El predicado llama a un predicado auxiliar que obtiene el elemento (el par) del Acorde en la posición que diga Inversión, suma doce a las notas menores a la que encuentra, y les pone a todos la duración del par que nos ha dado el predicado auxiliar. Después, los ordena aplicando el algoritmo quick_sort, de modo que el par que estaba en la posición Inversión quedará el primero.

⇨Ejemplo: inversion(1,[(2,2),(1,1),(5,5)],L).

L = [(2, 2), (5, 2), (13, 2)]

Retrogradar:

Este predicado recibe una lista con elementos(notamidi,duración) que puede contener listas anidadas (acordes que hay que procesar) o dosillos y tresillos y devuelve la lista al revés. Su sintaxis es la siguiente:

retrogradar(ListaEntrada, ListaSalida).

⇨Ejemplo:retrogradar([(60,8),(64,8),[(61,8),(62,8)],(67,8)],Lout)

Lout=[(67,8),[(61,8),(62,8)],(64,8),(60,8)]

⇨Ejemplo con un dosillo: retrogradar([(2,2),(1,1),'d',(3,3),(4,3),'f',(5,3)].L).
(el dosillo lo forman (3,3),(4,3))

L= [(5,3),'d',(4,3),(3,3),'f',(1,1),(2,2)]

Transponer:

Transponer recibe una lista que puede contener listas anidadas (acordes que hay que procesar) o dosillos y tresillos y devuelve la lista al revés. Su sintaxis es la siguiente:

transponer(intervalo, ListaEntrada, ListaSalida).

Funcionamiento: Le suma el valor de intervalo (puede ser negativo) a las notas que se le pasan en L.

⇨Ejemplo: transponer (2, [(60,8), (64,8), (67,8)]), L2).

L2= [(62,8),(66,8),(69,8)]

⇨Ejemplo: transponer (2, [(60,8), (64,8), [(61,4), (62,4)], (67,8)]),L2).

L2= [(62,8),(66,8),[(63,4),(65,4)],(69,8)]

Fusionar:

Este predicado recibe simplemente como entrada la lista a modificar, y la devuelve formando un acorde según se explica más abajo. Su sintaxis es la siguiente:

fusionar(ListaEntrada,ListaSalida)

Funcionamiento: Fusiona los elementos de la lista de entrada en un acorde, con la duración igual a la suma de las duraciones de las notas fusionadas. Si hay dosillos o tresillos, los trata como un bloque que no puede fusionarse.

⇨Ejemplo: fusionar([(1,2),(2,3),(3,1)], Lout).

Lout= [[(1,6), (2,6), (3,6)]].

⇨Ejemplo: Si hay un acorde en la lista de entrada quedaría:
fusionar([(1,2),(2,3),(3,1),[(4,2),(5,2),(6,2)]], Lout).

Lout=[[(1,8),(2,8),(3,8),(4,8),(5,8),(6,8)]].

Arpegiar:

Esta regla recibe como entrada la lista a modificar, formada por un acorde. Como salida devolverá una lista a la que ha aplicado la regla musical arpegiar. Su sintaxis es la siguiente:

arpegiar(Acorde,ListaNotas).

Funcionamiento: La lista de entrada es un acorde, la salida son las notas del acorde con cada nota con una duración igual a duración del acorde dividido entre el número de notas del acorde si la división es exacta. Si no lo es:

Mira si el resto de duración entre el número de elementos más uno es cero. Si es cero la duración de la notas arpegiadas es el resultado de la división, y pone uno más de los elementos aleatoriamente. Si no lo es:

Mira si el resto de duración entre el número de elementos menos uno es cero. Si es cero la duración de la notas arpegiadas es el resultado de la división, y hay que hacer fusionar2 con los dos últimos elementos.

Si no es cero realiza la misma operación con el número de elementos menos dos. Y si tampoco da exacta esta división, prueba entre el número de elementos menos tres.

Si en la entrada hay notas que no están incluidas en ningún acorde, simplemente las pasa y las deja igual que en la entrada.

⇨Ejemplo: arpegiar([[(1,3),(2,3),(3,3)]],Lout).
Lout=[(1,1),(2,1),(3,1)]

⇨Ejemplo: arpegiar([[(1,3),(2,2)] ,(1,2),(3,2), [(4,3),(5,3),(6,3)],s,[(1,2),(2,2)]] , L).
L=[(1,3),(2,2),(1,2),(3,2), (4,1),(5,1),(6,1),s,(1,1),(2,1)]

Cambio Dirección:

Esta regla recibe como entrada la lista a modificar, pudiendo contener cualquiera de las construcciones explicadas de nuestra aplicación. Su sintaxis es la siguiente:

cambioDir (ListaIn, ListaOut).

Funcionamiento: Tomando como entrada una lista de notas, toma como referencia la primera nota que pasa igual a la salida, el resto se transforman de la siguiente manera:

NotaI → NuevaNotaI= Nota1+(Nota1-NotaI)

⇨Ejemplo: cambioDir([(10,2),(8,3),(11,3),(12,2)], ListaOut).

ListaOut=[(10,2),(12,3),(9,3),(8,2)]

Expandir Semitonos:

Esta regla recibe como entrada la lista a modificar, pudiendo contener cualquiera de las construcciones de nuestra aplicación. Su sintaxis es la siguiente:

expandirSemitonos(Intervalo, ListaIn, ListaOut).

Funcionamiento: Realiza una operación similar a la de transponer pero añadiendo las nuevas notas a lo existente (fusionando).Pueden llegar acordes en la lista, en ese caso toma como referencia para la suma la última nota del acorde.

⇨Ejemplo: expandirSemitonos(2, [(1,2),(2,2),(3,4),(4,4)], ListaOut).

ListaOut =[[(1,2),(3,2)], [(2,2),(4,2)], [(3,4),(5,4)], [(4,4),(6,4)]]

⇨Ejemplo: expandirSemitonos(2,[[(1,2),(2,2),(3,2)],(4,4)], ListaOut).

ListaOut =[[(1,2),(2,2),(3,2),(5,2)],[(4,4),(6,4)]]

⇨Ejemplo:Se tratan las notas de los dosillos y tresillos normalmente.

expandirSemitonos(1,[(1,1),'d',(2,2),(3,2),'f',(4,4)],L).

L= [[(1,1),(2,1)],'d',[(2,2),(3,2)],[(3,2),(4,2)],'f',[(4,4),(5,4)]]

Expandir Intervalo:

Este predicado es parecido a expandirSemitonos pero este tiene en cuenta la escala. Su sintaxis es:

expandirIntervalo(Tonalidad, TipoEscala, Parametro, LIn, LOut).

Funcionamiento: añade el intervalo pasado como parámetro a las notas de la lista de entrada, es decir, mira si se encuentran en la escala definida por Tonalidad y TipoEscala, si la nota está, halla su posición y calcula la posición de la nueva nota, armonizada en función del parámetro de tipo entero

Parámetro. Entonces añade a la lista de salida un acorde formado por la nueva nota y la antigua, ambas con la misma duración, la de la nota antigua, de la lista de entrada. Si el parámetro es negativo, pone la nueva nota delante de la vieja en vez de detrás.

⇨Ejemplo: expandirIntervalo (0,0,2,[(9,1),[(0,1),(2,1),(4,1)],s,(5,1)], LOut).

LOut = [[(9,1),(12,1)], [(0,1),(2,1),(4,1),(7,1)], s, [(7,1),(11,1)]].

⇨Ejemplo: expandirIntervalo (0,0,-1,[(9,1),[(2,1),(4,1)],s,(5,1)], LOut).

LOut = [[(7, 1), (9, 1)], [(2, 1), (4, 1), (0, 1)], s, [(4, 1), (5, 1)]]

Progresión Cromática:

Esta regla recibe un entero que indica el número de progresiones que hay que realizar, una lista de entrada sobre la que aplicar las progresiones y devuelve la lista modificada en su tercer parámetro. Su sintaxis es:

progresionCromatica (NumProgresiones, ListaEntrada, ListaSalida).

Funcionamiento: En la salida se obtiene la lsita de entrada, y la lista repetida tantas veces como diga el parámetro NumProgresiones, pero sumando en cada iteración uno a cada nota.

⇨Ejemplo: progresionCromatica(2, [(1,1),[(2,1),(3,1)],s,(4,1)], LOut).

LOut=[(1,1),[(2,1),(3,1)],s,(4,1),s,(2,1),[(3,1),(4,1)],s,(5,1),s,(3,1),[(4,1),(5,1)],s,(6,1)]

Cambiar octava:

Predicado que recibe dos elementos de entrada, un entero y una lista y devuelve una lista de salida. Su sintaxis es:

cambiarOctava(ParamOctava, LIn, Lout).

Funcionamiento: Aumenta o disminuye la octava de las notas de la lista de entrada en función del parámetro ParamOctava, que puede ser negativo para disminuir o positivo para aumentar de octava las notas.

⇨Ejemplo: cambiarOctava(2, [(3,1),[(4,1),(5,1)],s,(6,1)], LOut).

LOut =[(27,1),[(28,1),(29,1)],s,(30,1)]

⇨Ejemplo: cambiarOctava(-1, [(15,1),[(16,1),(17,1)],s,(18,1)], LOut).

LOut =[(3,1),[(4,1),(5,1)],s,(6,1)]

Aumentar:

Este predicado se usa para aumentar la duración de las notas de la lista de entrada. Recibe como argumentos el número por el que multiplicar sus notas, el tipo de compás, la lista a modificar y devuelve otra lista con las duraciones aumentadas. Su sintaxis es:

aumentar(P, compas, Lin, Lout).

Funcionamiento: Aumenta la duración de los pares de la lista multiplicando por dos o por tres en función del valor de P. Primero debe llamar a normaliza(Lin, ListaNormalizada), predicado auxiliar que elimina las marcas de ligado, fin de compás, dosillos, etcétera para poder tratar la lista primero en reglas que afectan a la duración de las notas y después volver a poner las marcas en donde corresponda. Con la lista ListaNormalizada va rellenando los compases que hagan falta. Una cosa importante es que los 1000 no aumentan en duración sino en número.

⇨Ejemplo: aumentar(2,2-4, [(60,2),(1000,1),(1000,1), [(60,4),(64,4)], (1001,0),s, [(60,3),(64,3)],(67,5)],L).

L= [(60,4),(1000,1),(1000,1),(1000,1),(1000,1),s,
[(60,8),(64,8)],(1001,0),s,[(60,6),(64,6)],(67,2),(1001),s,(67,8)]

Trasladar:

La regla trasladar mete silencios al comienzo de la lista de entrada, para ello recibe además de la lista un parámetro P que indicará el número de silencios y el tipo de compás de la lista. Devolverá la lista modificada en su cuarto parámetro. La sintaxis del predicado es:

trasladar(P,compas,Lin,Lout).

Funcionamiento: Traslada los pares metiendo tantos silencios (1000,1) como indique P (entre 1 y 16). Primero habrá que normalizar Lin.

⇨Ejemplo: trasladar(2,2-4, [(60,2),(1000,1),(1000,1), [(60,4),(64,4)], (1001,0),s, [(60,3),(64,3)],(67,5)], L).

L=[(1000,1),(1000,1),(60,2),(1000,1),(1000,1),[(60,2),(64,2)],(1001,0),s,
[(60,5),(64,5)],(67,3),(1001,1),s,(67,2),(1000,1),(1000,1),(1000,1),(1000,1
), (1000,1),(1000,1)]

Disminuir y Repetir:

Este predicado modifica la lista de entrada poniendo todas sus notas con la mínima duración posible. Su sintaxis es:

disminuirYrepetir(Lin,Lout).

Funcionamiento: Suma las duraciones de la lista de entrada y después rellena la lista de salida alternando las notas de entrada con duración 1 hasta que se llegue a la duración total de la lista de entrada. Esta regla se puede aplicar con notas sueltas. Si recibe acordes debe fallar.

⇨Ejemplo: disminiurYrepetir([(60,3),(62,2)],L).

L=[(60,1),(62,1)(60,1),(62,1),(60,1)]

Dividir:

Esta regla es la opuesta a aumentar. Sus parámetros son:
-Divisor=parámetro que será 2 o 3 y dividirá por ese parámetro cuando sea posible y sino probará con el otro
-Compas= Indica el tipo de compas que usamos
-ListaEntrada= Lista que será modificada por el predicado
-ListaSalida= Lista que se obtiene como respuesta

Su sintaxis es:

dividir(Divisor, Compas, ListaEntrada, ListaSalida).

Funcionamiento: Divide la duración de las notas de la lista de entrada entre el parámetro Divisor. Los pares que tengan duración 1 se dejan como están.

⇨Ejemplo: dividir(2,2-4,[(60,2),(64,3)],L)

L=[(60,1),(60,1),(64,2),(64,1)]

Como al dividir 3 entre 2 no queda exacto, damos la duración 2 al primer par y uno al segundo.

⇨Ejemplo: dividir(2,2-4,[(60,8)s, [(62,4),(66,4)]],L)

L=[(60,4),(60,4)s,[(62,2),(66,2)], [(62,2),(66,2)]]

Ligar:

Este predicado recibe una lista que devolverá con los respectivos ligados entre sus notas o acordes cuando se de el caso de que sean iguales. Su sintaxis es:

ligar(ListaEntrada, ListaSalida).

Funcionamiento: Básicamente si las notas o el acorde son iguales los une en uno de duración igual la suma de las duraciones de los anteriores. Puede ligar entre sí dos o más notas o acordes.

⇨Ejemplo: ligar([(60,8), s, (60,8), s, (60,8)], L).

L=[(60,8), (1001,0), s, (60,8), (1001,0), s, (60,8)]

⇨Ejemplo: ligar([(1,1), (1,3), (1,2), (3,2), [(2,2) ,(3,2)], [(2,2), (3,2)], s, (5,1), (6,1), (6,1)], L).

L=[(1,6), (3,2), [(2,4), (3,4)], s, (5,1), (6,2)]

Octavar:

Regla que recibe una lista de entrada a la que aplica un procedimiento al que se ha decidido llamar octavar, devolviéndola "octavada" en una lista de salida. Su sintaxis es:

octavar(Lin,Lout).

Funcionamiento: No se aplica sobre acordes. Para cada par de la lista de entrada añade a la salida el para de entrada con duración la mitad y un par en la que la nota se obtiene sumando 12 (la octava superior de la de la entrada) con duración la mitad.

⇨Ejemplo: octavar ([(11,2), (22,2), (44,4)], Lout).

Lout = [(11, 1), (23, 1), (22, 1), (34, 1), (44, 2), (56, 2)]

Modular Escala:

Esta regla modula las notas de su lista de entrada en función de si pertenecen a una escala determinada por sus variables Tonalidad y EscalaOrigen. Además, tiene otro parámetro además de la lista de entrada y de salida, este a su vez informa de cual es la nueva escala para las notas. Su sintaxis es:

modularEscala(Tonalidad, EscalaOrigen, EscalaNueva, ListaIn, ListaOut).

Funcionamiento: Procesa la ListaIn de la siguiente forma: Vemos si la nota que estamos procesando está en escalaArmonía(Tonalidad,EscalaOrigen, ListaEscala). Si pertenece a ListaEscala obtenemos su posición y añadimos a la lista la nota que corresponde a esa posición pero en la escala EscalaNueva. Si no estaba añadimos la nota directamente a la salida.

⇨Ejemplo: modularEscala(0,0,8,[(0,2),(1,2),(4,2),(9,2)],ListaOut).

Para que el ejemplo quede más claro se han incluido los predicados auxiliares llamados escalaArmonía que utiliza:

escalaArmonia(0,0,[-10,-8,-7,-5,-3,-1,0,2,4,5,7,9,11,12,14,16,17,19,21]).
escalaArmonia(0,8,[-10,-8,-7,-5,-4,-1,0,2,4,5,7,8,11,12,14,16]).

(0, _) está en escalaArmonía(0,0,L) → posición 7, en escalaArmonía(0,8,) es la nota 0

(1,)No está en escalaArmonía(0,0,L) → se añade a la salida directamente

(2, _) está en escalaArmonía(0,0,L) → posición 8, en escalaArmonía(0,8,) es la nota 2

(4, _) está en escalaArmonía(0,0,L) → posición 9, en escalaArmonía(0,8,) es la nota 4

(9, _) está en escalaArmonía(0,0,L) → posición 12, en escalaArmonía(0,8,) es la nota 8

Lout queda: [(0,2),(1,2),(4,2),(8,2)]

Marcha Tonal:

Este predicado recibe al igual que el anterior dos parámetros para poder identificar si una nota pertenece o no a una escala, los parámetros Tonalidad y Escala. Param es un entero que nos dice el número de iteraciones que tendrá el predicado. La sintaxis de la regla es:

marchaTonal(Tonalidad,Escala,Param,ListaIn,ListaOut).

Funcionamiento: Va procesando las notas de ListaIn de la siguiente forma: Vemos si la nota que estamos procesando está en escalaArmonía(Tonalidad,Escala,Lista). Si está obtenemos su posición y añadimos a la lista de salida la nota que corresponde a (posición+p) en la Escala si param >0, y (posicion-p) si param<0. Si no estaba añadimos la nota+2 directamente a la salida.

⇨Ejemplo: marchaTonal(0,0,3,[(4,2),(6,2),s,(7,1)],ListaOut).

Tres iteraciones porque [param]=3

Iteración1:

(4, _) está en escalaArmonía(0,0,) → posición 9, se añade la nota de escalaArmonía que esté en (posición+1)

(6,)No está en escalaArmonía(0,0,) → se añade (8,) a la salida (la nota + (2*iteración))

(7, _) está en escalaArmonía(0,0,) → posición 11, se añade la nota de escalaArmonía que esté en (posición+1)

Iteración2:

(4, _)está en escalaArmonía(0,0,) → posición 9, se añade la nota en de la escalaArmonía que esté en (posición+2)

(6,)No está en escalaArmonía(0,0,) → se añade (10,) a la salida (la nota +(2*iteración))

(7, _) está en escalaArmonía(0,0,) → posición 11, se añade la nota en de la escalaArmonía que esté en (posición+2)

Iteración3:

(4, _) está en escalaArmonía(0,0,) → posición 9, se añade la nota en de la escalaArmonía que esté en (posición+3)

(6,)No está en escalaArmonía(0,0,) → se añade (12,) a la salida (la nota +(2*iteración))

(7, _) está en escalaArmonía(0,0,) → posición 11, se añade la nota en de la escalaArmonía que esté en(posición+3)

LOut=[(4,2),(6,2),s,(7,1),s,(5,2),(8,2),s,(9,1),s,(7,2),(10,2),s,(11,1)]

5.Herramientas empleadas

Para la interfaz y la pianola usamos java. Hemos elegido este lenguaje de programación por varios motivos:

- Teníamos experiencia en el desarrollo de aplicaciones java.
- Nos ofrece facilidades para desarrollar interfaces gráficas de usuario.
- Es un sistema multiplataforma que nos permite hacer versiones para distintos sistemas operativos con cambios mínimos.
- La existencia de las librerías de tuprolog nos permitía integrar fácilmente el código prolog en la interfaz.

Para la implementación de las reglas de generación y modificación de fragmentos musicales escogimos Prolog porque teníamos cierta experiencia con el lenguaje, las reglas s se adaptaban con bastante facilidad a la recursividad y nos permitía escribir código con bastante rapidez y por lo tanto probar ejemplos rápidamente.

5.1.Herramientas de desarrollo

- Jbuilder 2005 Foundation de uso libre. Ha sido la aplicación en la que hemos construido la interfaz y la pianola y todo el código java.

- SWI-PROLOG: empleado para desarrollar los procedimientos compositivos en prolog, aunque Orfeo no lo utiliza a la hora de ejecutar el código escrito en prolog.

- Tuprolog: librería java que permite integrar archivos de prolog en una aplicación java. Estas librerías emulan el comportamiento de prolog.

5.2.Herramientas integradas

- Abc2midi: aplicación que permite crear un archivo midi a partir de un archivo en notación ABC.

- Abcm2ps: aplicación que permite crear un archivo ps a partir de un archivo en notación ABC, para representar como partitura el contenido del archivo ABC.

- Timidity: es un sintetizador software que nos ha permitido reproducir la música creada con Orfeo. Ofrece la posibilidad de elegir las fuentes de sonido y gran número de opciones de reproducción.

- Ghostscript: es el programa intérprete por excelencia de documentos en formato. Ghostscript permite presentar datos PS en la pantalla y además traducirlos de manera que puedan ser impresos en una impresora con capacidad gráfica mediante el uso del controlador de dicha impresora.

- Gsview32: es un visor de libre distribución para archivos PS. Usado junto a Ghostscript y abcm2ps nos permite ver en partitura la música generada con Orfeo.

Todas las herramientas utilizadas son de libre distribución.

6.Principales dificultades halladas.

Problemas relacionados con la comunicación entre la interfaz y la Pianola:

Uno de los momentos más críticos en el desarrollo de la aplicación ha sido el de comunicar eficazmente la interfaz con la pianola. El problema a resolver era cómo poder expresar el contenido en formato ABC en la pianola, sin perder información y cómo recuperar la información de la pianola para expresarla posteriormente en formato ABC. El problema se solventó creando nuevas estructuras de datos que nos permitieron almacenar toda la información necesaria.

Problemas relacionados con el entorno de desarrollo SWI-Prolog:

Uno de los problemas más inquietantes con los que nos encontramos al comenzar a planificar nuestro proyecto fue el de la compatibilidad de códigos. Por suerte el código del entorno de desarrollo SWI-Prolog se adaptó con bastante comodidad, salvo algunas llamadas que no tenían su equivalente en el parser que utilizamos, TuProlog, del que hablaremos en su sección.

Otro inconveniente que planteaba Prolog era la notación, nuestra aplicación usa una versión de la notación musical ABC, pero Prolog debía recibir cifras numéricas para poder tratar con ellas con facilidad. De este modo, optamos por definir las notas en Prolog con cifras numéricas, de modo totalmente transparente al usuario.

Derivando de este problema, se dio un problema al tratar las marcas de final de compás, al tratar silencios, al hacer ligaduras entre notas de diferentes compases, etcétera. Para solventarlo fuimos definiéndonos caracteres reservados, (como se puede ser en la sección Sintaxis Utilizada y Estructuras de Datos) de modo que quedaran perfectamente definidos. Tras algunas complicaciones iniciales, la notación se estabilizó y se pudo desarrollar código con normalidad.

Problemas relacionados con la herramienta tuProlog:

Aunque tuProlog acepta un código muy semejante al que se genera en SWI-Prolog, no es exactamente el mismo, y como es tuProlog el encargado de que Java consiga las respuestas correctas, se debe tener en cuenta que lo que se está usando realmente para llevar a cabo el desarrollo de la parte lógico-algorítmica es código compatible con tuProlog, que además lo es con SWI-Prolog, que es el entorno usado para realizar todo tipo de pruebas y modificaciones. Pero debemos ajustarnos a los parámetros que nos marque tuProlog, sino, aunque nuestro programa funcione sobre SWI-Prolog, puede que no lo haga al integrarlo con Java.

Esta peculiaridad nos ha obligado a tener que desechar algunas funciones predefinidas en SWI-Prolog y definir unas propias o bien usar librerías de tuProlog que contuvieran los predicados que necesitábamos.

El caso de length(Lista, Longitud) es uno de los primeros. Lo utilizamos en diversas operaciones de nuestra aplicación, en principio llamando a la función predefinida, pero tuProlog no la aceptaba, por lo que tuvimos que definir una nueva. Algo parecido ocurrió con otras funciones auxiliares sencillas parecidas al caso comentado, como dameElem(Lista, Pos, Elemento), que es la función definida por nosotros para obtener el Elemento que se halle en la posición Pos de Lista para no usar el predefinido.

Por otro lado, también hubo funciones, como random(N) o el procedimiento de ordenación quicksort, en que tuvimos que recurrir a buscar librerías de tuProlog que permitieran su uso.

Sin embargo, en líneas generales tuProlog ha satisfecho con creces nuestra necesidad de hacer de nexo entre nuestros entornos de programación sin conllevar una excesiva necesidad de adaptación de nuestro sistema y de su código.

7.Objetivos alcanzados y posibles ampliaciones

7.1. Objetivos alcanzados

Al principio teníamos bastante claro las funcionalidades que debía tener Orfeo pero no teníamos claro cómo las íbamos a desarrollar y cómo se usarían.

El objetivo de la aplicación era ayudar al compositor en labores tediosas y hacer posible la escritura rápida de ideas musicales. Este objetivo ha sido cumplido. Era fundamental crear un entorno que permitiera un uso fácil e intuitivo y que ofreciera la posibilidad de ver y escucharen todo momento los música que se generaba. Estos objetivos han sido conseguidos.

Las propuestas del profesor han sido fundamentales a la hora de obtener una aplicación que fuera fácil de usar y que permitiera un grado elevado interacción y configuración, haciendo posible que cada usuario pueda crear sus propias "librerías" de progresiones, ritmos, percusiones y acordes.

7.2. Posibles ampliaciones

Algunas ampliaciones y mejoras posibles son:

- Añadir más opciones como la posibilidad de más tipos de compases (configurables desde archivo), soporte para más grupos a parte de dosillos y tresillos, capacidad para incluir instrucciones de dinámica y notas de adorno, pianola que permita trabajar con grupos, pianola con más escalas que puedan ser cargadas desde archivo.

- Añadir aleatoriedad en las opciones de generar y la opción de aplicar procedimientos aleatorios al fragmento seleccionado.

- Integrar sonidos de un sintetizador de mayor calidad que los que se obtienen con midi estándar.

- Integrar Orfeo y Genaro, añadiendo de esta forma la posibilidad de armonizar una melodía automáticamente.

- Introducir un nuevo lenguaje que permita generar obras con reglas que definan la forma y estructura de la composición con un nivel de abstracción mayor que el disponible actualmente.

- Versión para entornos Linux. Se puede obtener sin demasiados cambios debido a que las aplicaciones java son multiplataforma. Simplemente deberíamos hacer modificaciones en las llamadas a los programas auxiliares para que ejecutaran aplicaciones de Linux. Se distribuirían dos versiones de Orfeo que sólo se diferenciarían por los programas auxiliares. Existen versiones de timidity y gsview para linux.

8.Organización del proyecto

Desde un principio tuvimos claro que la aplicación podía dividirse en tres áreas de trabajo con cierta independencia que facilitaba el reparto de tareas entre los tres miembros del grupo.

Las tres áreas de trabajo han sido:

-Desarrollo de la interfaz principal.

-Desarrollo de la pianola.

-Programación de los procedimientos compositivos usando prolog.

La existencia de áreas diferenciadas nos ha permitido usar lenguajes de programación diferentes para áreas distintas sin ocasionar problemas.

Esta división nos ha permitido avanzar aún cuando algún área de trabajo tenía problemas y ha ahorrado reuniones presenciales entre los miembros del grupo que debido a los distintos horarios académicos y laborales que teníamos han sido difíciles de realizar.

Hemos aprovechado al máximo el correo electrónico convirtiéndose en el verdadero medio de comunicación y transmisión de código y documentación entre los integrantes del proyecto.

El encargado de desarrollar la interfaz principal mantenía siempre la última versión de la aplicación con las tres áreas de trabajo integradas. De esta manera no convivían versiones integradas en las que cada parte fuera de una versión distinta. Las versiones podían ser fácilmente identificadas porque incluíamos la fecha en el nombre de cada versión.

9.Apéndices

9.1.Mapa de instrumentos General Midi:

Coincide con los instrumentos disponibles en Orfeo.

0	Piano de cola	1	Piano brillante	2	Piano de cola eléctrico	3	Piano de bar
4	Piano eléctrico Rhodes	5	Piano eléctrico con chorus	6	Clavicordio	7	Clavinet
8	Celesta	9	Glockenspiel	10	Caja de música	11	Vibráfono
12	Marimba	13	Xilófono	14	Campanas tubulares	15	Salterio
16	Organo Hammond	17	Organo percusivo	18	Organo de rock	19	Organo de iglesia
20	Armonio	21	Acordeón	22	Armónica	23	Bandoneón
24	Guitarra española	25	Guitarra acústica	26	Guitarra eléctrica de jazz	27	Guitarra eléctrica limpia
28	Guitarra eléctrica apagada	29	Guitarra eléctrica con overdrive	30	Guitarra eléctrica distorsionada	31	Armónicos de guitarra eléctrica
32	Bajo acústico	33	Bajo eléctrico (dedos)	34	Bajo eléctrico (púa)	35	Bajo eléctrico sin trastes
36	Bajo eléctrico golpeado 1	37	Bajo eléctrico golpeado 2	38	Bajo sintético1	39	Bajo sintético2
40	Violín	41	Viola	42	Violoncelo	43	Contrabajo
44	Violín trémolo	45	Violín pizzicato	46	Arpa	47	Timbal de orquesta
48	Sección de cuerda 1	49	Sección de cuerda 2	50	Sección de cuerda sintética 1	51	Sección de cuerda sintética 2
52	Coro Aahs	53	Coro Oohs	54	Voz sintética	55	Golpe de orquesta
56	Trompeta	57	Trombón	58	Tuba	59	Trompeta con sordina
60	Fiscorno	61	Sección de metal	62	Sección de metal sintética 1	63	Sección de metal sintética 2
64	Saxo soprano	65	Saxo alto	66	Saxo Tenor	67	Saxo Barítono
68	Oboe	69	Corno inglés	70	Fagot	71	Clarinete
72	Flautín	73	Flauta travesera	74	Flauta de pico	75	Flauta de Pan
76	Cuello de botella (soplo)	77	Shakuhachi (flauta japonesa)	78	Silbido	79	Ocarina
80	Sinte melodía 1 (onda cuadrada)	81	Sinte melodía 2 (diente sierra)	82	Sinte melodía 3 (órgano)	83	Sinte melodía 4 (siseo)
84	Sinte melodía 5 (charango)	85	Sinte melodía 6 (vocal)	86	Sinte melodía 7 (quintas)	87	Sinte melodía 8 (bajo)
88	Sinte armonía 1 (new age)	89	Sinte armonía 2 (cálido)	90	Sinte armonía 3 (polysynth)	91	Sinte armonía 4 (Coral)
92	Sinte armonía 5 (arco)	93	Sinte armonía 6 (metálico)	94	Sinte armonía 7 (celestial)	95	Sinte armonía 8 (filtro)
96	Sinte efecto 1 (lluvia)	97	Sinte efecto 2 (banda sonora)	98	Sinte efecto 3 (cristales)	99	Sinte efecto 4 (atmosférico)
100	Sinte efecto 5 (brillante)	101	Sinte efecto 6 (duendes)	102	Sinte efecto 7 (ecos)	103	Sinte efecto 8 (ciencia ficción)
104	Sitar	105	Banjo	106	Shamisen (laúd japonés)	107	Koto (cítara japonesa)
108	Kalimba	109	Gaita	110	Violín country	111	Shannai (dulzaina hindú)
112	Cascabeles	113	Agogó	114	Steel Drum	115	Caja de madera
116	Taiko (tambor japonés)	117	Timbal melódico	118	Caja sintética	119	Platillo invertido
120	Trasteo de guitarra	121	Respiración	122	Orilla del mar	123	Trino
124	Timbre de teléfono	125	Helicóptero	126	Aplausos	127	Disparo

9.2. Mapa General Midi de instrumentos de percusión:

27	High Q	28	Slap	29	Scratch Push	30	Scratch Pull
31	Sticks	32	Square Click	33	Metronome Click	34	Metronome Bell
35	Bass Drum 2	36	Bass Drum 1	37	Side Stick	38	Acoustic Snare
39	Handclap	40	Electric Snare	41	Low Floor Tom	42	Closed High Hat
43	High Floor Tom	44	Pedal High Hat	45	Low Tom	46	High Hat
47	Low Mid Tom	48	High Mid Tom	49	Crash Cymbal 1	50	High Tom
51	Ride Cymbal 1	52	Chinese Cymbal	53	Ride Bell	54	Tambourine
55	Splash Cymbal	56	Cowbell	57	Crash Cymbal 2	58	Vibraslap
59	Ride Cymbal 2	60	High Bongo	61	Low Bongo	62	Mute High Conga
63	High Conga	64	Low Conga	65	High Timbale	66	Low Timbale
67	High Agogo	68	Low Agogo	69	Cabasa	70	Maracas
71	Short Whistle	72	Long Whistle	73	Short Guiro	74	Long Guiro
75	Claves	76	High Wood Block	77	Low Wood Block	78	Mute Cuica
79	Cuica	80	Mute Triangle	81	Triangle	82	Shaker
83	Jingle Bell	84	Belltree	85	Castanets	86	Mute Surdo
87	Open Surdo						

9.3. Kits de Percusión General Standard

Programa	Nombre	Descripción
1	Standard	
9	Room	menos reverberación
17	Power	más contundente
25	Electronic	electrónica analógica
26	TR-808	electrónica típica techno
33	Jazz	muy similar a la standard
41	Brush	escobillas
49	Orchestra	percusión de orquesta

9.4. Relación de notas MIDI y frecuencias

Nota	Freq. (Hz)	MIDI	Nota	Freq.(Hz)	MIDI	Nota	Freq.(Hz)	MIDI
La 1	27.500	21	La 4	220.000	57	La 7	1760.000	93
La# 1	29.135	22	La# 4	233.082	58	La# 7	1864.655	94
Si 1	30.868	23	Si 4	246.942	59	Si 7	1975.533	95
Do 2	32.703	24	Do 5	261.626	60	Do 8	2093.005	96
Do# 2	34.648	25	Do# 5	277.183	61	Do# 8	2217.461	97
Re 2	36.708	26	Re 5	293.665	62	Re 8	2349.318	98
Re# 2	38.891	27	Re# 5	311.127	63	Re# 8	2489.016	99
Mi 2	41.203	28	Mi 5	329.628	64	Mi 8	2637.021	100
Fa 2	43.654	29	Fa 5	349.228	65	Fa 8	2793.826	101
Fa# 2	46.249	30	Fa# 5	369.994	66	Fa# 8	2959.956	102
Sol 2	48.999	31	Sol 5	391.995	67	Sol 8	3135.964	103
Sol# 2	51.913	32	Sol# 5	415.305	68	Sol# 8	3322.438	104
La 2	55.000	33	La 5	440.000	69	La 8	3520.000	105
La# 2	58.270	34	La# 5	466.164	70	La# 8	3729.310	106
Si 2	61.735	35	Si 5	493.883	71	Si 8	3951.066	107
Do 3	65.406	36	Do 6	523.251	72	Do 9	4186.009	108
Do# 3	69.296	37	Do# 6	554.365	73	Do# 9	4434.922	109
Re 3	73.416	38	Re 6	587.330	74	Re 9	4698.637	110
Re# 3	77.782	39	Re# 6	622.254	75	Re# 9	4978.032	111
Mi 3	82.407	40	Mi 6	659.255	76	Mi 9	5274.042	112
Fa 3	87.307	41	Fa 6	698.457	77	Fa 9	5587.652	113
Fa# 3	92.499	42	Fa# 6	739.989	78	Fa# 9	5919.912	114
Sol 3	97.999	43	Sol 6	783.991	79	Sol 9	6271.928	115
Sol# 3	103.826	44	Sol# 6	830.609	80	Sol# 9	6644.876	116
La 3	110.000	45	La 6	880.000	81	La 9	7040.000	117
La# 3	116.541	46	La# 6	932.328	82	La# 9	7458.620	118
Si 3	123.471	47	Si 6	987.767	83	Si 9	7902.133	119
Do 4	130.813	48	Do 7	1046.502	84	Do 10	8372.019	120
Do# 4	138.591	49	Do# 7	1108.731	85	Do# 10	8869.845	121
Re 4	146.832	50	Re 7	1174.659	86	Re 10	9397.273	122
Re# 4	155.564	51	Re# 7	1244.508	87	Re# 10	9956.064	123
Mi 4	164.814	52	Mi 7	1318.510	88	Mi 10	10548.083	124
Fa 4	174.614	53	Fa 7	1396.913	89	Fa 10	11175.305	125
Fa# 4	184.997	54	Fa# 7	1479.978	90	Fa# 10	11839.823	126
Sol 4	195.998	55	Sol 7	1567.982	91	Sol 10	12543.855	127
Sol# 4	207.652	56	Sol# 7	1661.219	92	Sol# 10	13289.752	128

10.BIBLIOGRAFÍA

♦Fundamentals of Musical Composition.
Autor: Arnold Schoenberg
Faber and Faber, 1967

♦Cábalas con la Guitarra
Autor: Gabriel Rosales
Fundación Autor

♦Atlas de música
Autor Ulrich Michels
Alianza Editorial

♦ Página del proyecto ABC
http://abc.sourceforge.net/

♦ Página del Proyecto ABCplus
http://abcplus.sourceforge.net/

♦ Completa página sobre ABC
http://www.walshaw.plus.com/abc/

♦ Tutorial de ABC
http://www.lesession.co.uk/abc/abc_notation.htm

♦ Página sobre audio digital y midi
http://www.ccapitalia.net/reso/articulos/audiodigital/

♦ Página de la asociación de los fabricantes de tecnología midi
http://www.midi.org/

♦ Página de Timidity
http://timidity.sourceforge.net/

♦ Página de Ghostscript y GSview
http://www.cs.wisc.edu/~ghost/

♦ Página de Sun con tutoriales de java
http://java.sun.com/docs/books/tutorial/

♦ http://java.sun.com/j2se/1.5.0/docs/api/

♦Thinking in Java
Autor: Bruce Eckel
Prentice-Hall

♦ Páginas relacionadas con la creación de interfaces gráficas en Java

http://www.cica.es/formacion/JavaTut/Cap4/barra.html/

http://www.programacion.net/java/tutorial/ags_j2me/5/

http://www.htmlpoint.com/guidajava/java_36.htm

http://www.rgagnon.com/javadetails/java-0230.html

♦ Clause and Effect: Prolog Programming for the Working Programmer
Autor: William S. Clocksin
Springer, cop. 1997

♦ Programming inProlog (Using The ISO Standard)
Autores W. S. Clocksin, C. S. Mellish
Springer, cop. 2003, 5th edición.

♦ La página web de JBuilder
http://www.codegear.com/products/jbuilder/

♦ La página web de tuProlog
http://www.alice.unibo.it/tuProlog/

♦ La página web de SWI-Prolog
http://www.swi-prolog.org/

Los abajo firmantes, expresan su conformidad a autorizar a la Universidad Complutense de Madrid, a difundir y utilizar con fines académicos, en ningún caso comerciales y mencionando expresamente a sus autores, tanto la presente memoria, como el código del proyecto, el prototipo desarrollado y la documentación que se considere necesaria.

Firmado Ignacio Ferrando García

Firmado Luis Molero Blázquez

Firmado Carlos Rioboo Crespo